Comparaison des RNN, LSTM et des GRU dans le cadre du pos-tagging de texte

Lien git du projet : https://github.com/noeleel/INF8225_RNNvsLSTM

NOELE Elodie, TABOGA Vincent, WOZNIAK Anne-Laure

elodie.noele@polymtl.ca
vincent.taboga@polymtl.ca
anne-laure.wozniak@polymtl.ca

Résumé

Ce rapport vise à étudier les performances de différents modèles de RNN, de GRU et de LSTM dans le cadre du pos-tagging de texte. Le pos-tagging de texte consiste à associer chaque mot d'un texte à une entité grammaticale bien définie. Parmi les structures testées, on retrouve les modèles vanilla RNN, RNN à deux couches et bidirectionnels, les LSTM simples, bidirectionnels et à plusieurs couches ainsi que les vanilla GRU, les GRU bidirectionnels et les GRU à deux couches. Nous avons aussi ajouté pour certaines architectures un Dropout standard (valant 0.5 i.e. on ne prend qu'une moitié du réseau à chaque itération durant l'entraînement). Les tests réalisés montrent que les LSTM et les GRU performent mieux pour le POS tagging, comme on pouvait si attendre car c'est une opération qui nécessite de mémoriser des séquences. Il n'y a cependant pas de différences significatives de performance entre les GRU et les LSTM.

Introduction

Le POS-Tagging de texte est une étape préliminaire à la traduction de texte d'une langue à l'autre. C'est une des tâches courantes du traitement du langage. Cette technique peut aussi être utilisée pour identifier les mots clés d'une phrase sur un moteur de recherche et ainsi mieux cibler la demande d'un utilisateur.

Le POS-Tagging est un domaine couramment étudié en intelligence artificielle, puisqu'il donne généralement de bien meilleurs résultats que le pos-tagging avec des algorithmes « classiques » comme celui proposé par la bibliothèque *NLTK* de Python.

Dans cet article, nous proposons différents types de réseaux neuronaux récurrents que nous évaluons sur une expérience de POS-tagging : Les vanilla RNN, les RNN à deux couches, les RNN bidirectionnel, les LSTM simples, les LSTM à deux couches, les LSTM bidirectionnel, les GRU simples, les GRU bidirectionnel et les GRU à deux couches. Nous évaluons ces réseaux le corpus de texte « Brown[1] » de la librairie *NLTK* de python.

L'article est donc organisé de la manière suivante : en premier lieu nous présentons quelques travaux antérieurs, puis nous explicitons les modèles retenus pour notre étude. Nous décrivons ensuite les expériences menées sur ces modèles ainsi que nos résultats avant de discuter des améliorations possibles de notre travail.

Travaux antérieurs

Les réseaux neuronaux récurrents sont les candidats parfaits pour du POS-Tagging car ils permettent de mémoriser des séquences en prenant en compte l'état des unités aux instants antérieurs et postérieurs. Le but initial de ce projet était de comparer les performances entre des réseaux récurrents classiques et des réseau Long Short Term Memory (LSTM). Cependant les travaux déjà réalisés dans ce domaine font états de l'utilisation de réseaux à « Gated Reccurent Unit » (GRU). Nous avons donc élargi le champ de notre recherche à ce troisième type d'architecture.

Les GRU montrent en effet les meilleures performances pour ce type d'application, au même titre que les LSTM malgré une architecture plus simple.

On notera toutefois qu'à ce jour l'architecture ayant obtenue le score le plus élevé est un LSTM bidirectionnel couplés à des méthodes statistiques comme le CRF, développé par **Wei Xu et al.** Les techniques employées dépassent cependant le cadre de cette étude. Nous nous limiterons ici à l'utilisation de réseaux neuronaux sans méthode statistique.

Modèles retenus

RNN

Les réseaux neuronaux récurrents sont des réseaux dont les connexions forment des boucles. Chaque neurone a ainsi en entrée l'état du système à l'instant t ainsi que les informations des cellules au temps t-1. Des tels réseaux

[1]Le Brown Corpus est un recueil de près de 500 extraits de textes issus de la littérature anglaise, contenant près d'un million de mots. Il a été compilé par Henry Kučera et W. Nelson Francis à Brown University, Providence, Rhode Island afin de servir dans le domaine de l'étude de texte. Il est encore très utilisé dans le cadre de l'intelligence artificielle pour entraîner les réseaux neuronaux appliqués au traitement du langage naturel.

possèdent un état interne, ce qui en font des candidats propices pour des tâches de reconnaissance séquentielle comme le traitement du langage naturel. Au sein d'un réseau des opérations matricielles sont appliqués sur l'observation du système (noté o) et sur les états cachées (noté h) de l'état passé.

Les résultats de ces opérations sont ensuite passés en entrée d'une fonction d'activation (ici log-softmax).

$$\boldsymbol{h_t = activation(W_h x_t + U_h h_{t-1} + b_t)}$$
$$\boldsymbol{o_t = activation(W_o h_t + b_o)}$$

Avec :

U_h, la matrice du réseau
W_h, la matrice des poids de l'unité cachée
W_o, la matrice des poids de la sortie
b_t, le terme de biais de l'unité cachée
b_o, le terme de biais de la sortie

LSTM

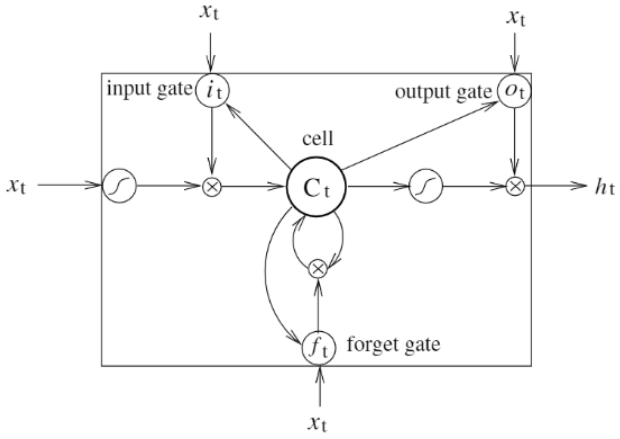

Figure 1: Représentation d'une cellule LSTM

Les réseau Long Short Term Memory (LSTM) sont des RNN dont les cellules ont une structure particulière. Nous avons vu précédemment que deux informations principales sont contenues dans une même variable au sein d'un réseau de neurones récurrents : la valeur retournée au réseau pour faire des prédictions et la séquence de données utilisées jusqu'ici. Une cellule LSTM va séparer stocker ces deux informations dans deux variables séparées : une variable de mémoire c et une variable de l'état de la cellule h. La mise à jour de ces variables se fait à l'aide de trois portes : une d'entrée, une de sortie et une de réinitialisation de la mémoire (*input gate, ouput gate* et *forget gate*). Une telle architecture est plus complexe mais présente de très bons résultats pour beaucoup d'applications. Elles permettent notamment de résoudre les problèmes de gradient qui explose ou qui disparait lors de la rétropropagation dans le temps.

L'information gardée en mémoire ne sera pas altérée au fur et à mesure des itérations. Une telle architecture est représentée sur la figure 1. Les fonctions classiquement utilisées pour les LSTM, et que nous avons retenues pour nos modèles, sont les suivantes :

$$\boldsymbol{h_t = \tanh(c_t)\ o\ o_t}$$
$$\boldsymbol{o_t = sigmoid(W_o x_t + U_o h_{t-1} + b_o)}$$
$$\boldsymbol{f_t = sigmoid(W_f x_t + U_f h_{t-1} + b_f)}$$
$$\boldsymbol{i_t = sigmoid(W_i x_t + U_i h_{t-1} + b_i)}$$
$$\boldsymbol{j_t = \tanh(W_x x_t + W_h h_t + b_j)}$$
$$\boldsymbol{y_t = softmax(W_h h_t + b_y)}$$
$$\boldsymbol{c_t = f_t\ o\ c_{t-1} + i_t\ o\ j_t}$$

Avec:

i: input gate
o: ouput gate
f : forget gate
j : new memory content
c : update memory

GRU

Les réseaux à « Gated Recurrent Unit » (GRU) sont basés sur le même principe que les LSTM. Ils ont cependant une architecture un peu plus simple car ils ne possèdent que deux portes : entrée et sortie. Il n'y a donc pas de moyen de contrôler l'information qui doit être gardé en mémoire. Ils permettent tout de même d'éviter les difficultés d'apprentissage liées à la disparition du gradient. En contrepartie de leur manque de flexibilité, ils sont plus faciles à entraîner car ils contiennent moins de paramètres. On notera de plus que le gain d'une troisième porte sur les performances n'est pas significatif pour beaucoup de tâches. Nous vérifierons plus tard si c'est le cas pour du POS tagging.

$$s_t = (1 - z)\ o\ h + z\ o\ s_{t-1}$$
$$h = tanh(x_t U_h + (r\ o\ s_{t-1}) W_h)$$
$$z = sigmoid(U_z x_t + W_z s_{t-1})$$
$$r = sigmoid(W_r s_{t-1} + U_r x_t)$$

Réseau à deux couches

Les réseaux à deux couches sont un type de réseau neuronal où l'information circule de la couche d'entrée, vers les couches cachées puis la couche de sortie. C'est un réseau de type feedforward. Comparé au réseau neuronal dit "simple", les réseaux à deux couches, couplé à la rétropropagation du gradient, permettent aux réseaux neuronaux d'effectuer des taches plus abstraites, comme l'exercice du POS-Tagging qui est un problème non linéaire.

Il aurait été possible d'utiliser des réseaux à plus de deux couches, nos expériences nous ont montré que les résultats n'étaient pas forcément meilleurs, notamment dans le cas des RNN où nous pourrions faire face à un problème de gradient explosif ou évanescent.

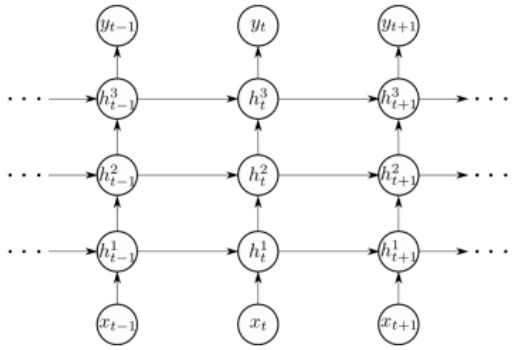

Figure 2 : Réseau RNN à 3 couches cachées : x l'entrée, h l'état de la couche cachée, y la sortie.

Réseau bidirectionnel

Dans le cadre du POS-tagging, il peut être bénéfique de savoir quels sont les mots avant et quels sont les mots après le mot que l'on souhaite classifier. C'est pourquoi l'utilisation des réseaux neuronaux bidirectionnels se prête volontiers à cette tâche.

Dans un réseau bidirectionnel, les informations passées et futures sont mémorisées dans deux sous-réseaux avant et arrière. Il fonctionne en trois étapes :

- Le réseau arrière est utilisé pour déterminer les tags et les états de la couche cachée (s'il y en a, par exemple pour les LSTM)
- Ensuite, ces tags et ces états de la couche caché sont utilisés comme données d'entrée pour le réseau avant.
- Enfin, le modèle global parcourt la séquence complète en utilisant les tags et les états des réseaux avant et arrière.

L'avantage de ce type de réseau est qu'il permet d'avoir une « compréhension » plus fine de la phrase à tagger et donc d'éviter les erreurs. Cette procédure a été définie dans l'article plus précisément dans l'article de de **Schuster & Paliwal, 1997**.

Expériences menées

Pour tester nos réseaux neuronaux récurrents, nous avons choisi d'utiliser la bibliothèque *NLTK* de Python qui propose le corpus de texte Brown dont le pos-tagging a déjà été réalisé. Ce corpus de texte possède plus de 50 000 phrases issues de la littérature anglaise. Nous avons donc fait le choix de tester nos modèles sur ces corpus en prenant **un ensemble d'entraînement de 5000 phrases**, **un ensemble de validation de 2000 phrases** et **un ensemble de tests de 1000 phrases**. Toutes ces phrases ont des longueurs aléatoires.

En outre, pour avoir une représentation plus fine de la performance de nos modèles nous avons choisi de tester nos modèles sur deux autres ensembles, le premier avec 1000 phrases courtes (moins de 5 entités grammaticales) et le second avec 1000 phrases longues (plus de 40 entités grammaticales). Nous pouvons ainsi comparer les performances de nos architectures selon que la séquence à mémoriser soit longue ou courte.

Conformément à ce qui est généralement fait en POS tagging, la performance sur les jeux de tests est évaluée avec le score f1, qui pénalises certaines erreurs plus que d'autres. Les résultats sont présentés dans le tableau si dessous :

Tableau 1 : F1 accuracy : phrases de taille aléatoire

Architecture	F1score Test set
Vanilla RNN	72,16 %
RNN à deux couches	73,31 %
RNN bidirectionnel	78,11 %
LSTM simples	82,03 %
LTSM à deux couches	85,02 %
LSTM bidirectionnel	85,12 %
LSTM à deux couches bidirectionnel	**85,88 %**
GRU simple	75,36 %
GRU à deux couches	80,92 %
GRU bidirectionnel	75,59 %

Tableau 2 : F1 accuracy : phrases de taille courte

Architecture	F1score Test set
LSTM simples	52,92 %
LTSM à deux couches	55,11 %
LSTM bidirectionnel	59,58 %
LSTM à deux couches bidirectionnel	**64,61 %**
GRU simple	53,13 %
GRU à deux couches	49,38 %
GRU bidirectionnel	53,37 %

Tableau 3 : F1 accuracy : phrases de taille longue

Architecture	F1score Test set
LSTM simples	84,04 %
LTSM à deux couches	84,44 %
LSTM bidirectionnel	86,83 %
LSTM à deux couches bidirectionnel	**87,73 %**
GRU simple	83,87 %
GRU à deux couches	82,00 %
GRU bidirectionnel	83,47 %

Si nous comparons nos résultats par classe de modèles, on peut retenir les RNN bidirectionnel avec un résultat de 78,11 %, les LSTM bidirectionnel à deux couches avec un résultat de 85,88%, les GRU à deux couches avec un résultat de 80,92%. Les réseaux bidirectionnels sont naturellement plus performants, car ils permettent une meilleure « compréhension » de l'environnement des mots au sein d'une phrase comme nous l'expliquions dans une des sections précédentes.

A complexité de réseaux équivalente, i.e. en ne considérant que les réseaux bidirectionnels, ou uniquement les réseaux à deux couches, etc., on s'aperçoit que les LSTM sont vraiment les plus performants, quelle que soit la catégorie.

Cependant, on peut noter que les résultats sont globalement plus bas avec des phrases courtes, chaque erreur est dans ce cas très pénalisante, d'où les faibles scores. L'écart de performance entre les GRU et les LSTM reste le même indépendamment de la longueur des phrases. L'influence de la « forget gate » semble être importante autant pour les courtes séquences que pour les longues.

Si on s'intéresse maintenant au nombre d'épisodes nécessaires pour entrainer les architectures, on remarque que les réseaux LSTM et GRU prennent sensiblement le même nombre d'épisode (entre 4 et 6) pour que les scores du jeu de validation convergent. Il faut plus d'épisodes au RNN classique, qui semble tout juste converger après 9 épisodes. La *Figure 3* représente l'évolution du score f1 sur le jeu de validation pour l'architecture de chaque type (RNN, GRU et LSTM)

Outre le score obtenu sur le jeu de test, une problématique importante est le temps d'entraînement des modèles. Des architectures complexes peuvent en effet mettre plusieurs semaines à s'entrainer sur des jeux de plusieurs milliers de données.

Le *Tableau 4* présente le temps mis par les architecteurs GRU et LSTM pour s'entraîner, c'est-à-dire à parcourir 10 fois le jeu de test et de validation. Les entraînements ont été réalisés sur la même machine pour pouvoir comparer les résultats. On notera toutefois que les valeurs sont à prendre relativement les unes aux autres. Elles dépendent en effet de la puissance de la machine ainsi que de l'efficacité du code. Nous n'avons par exemple pas utilisé de batch dans les épisodes, ce qui aurait réduit le temps d'entrainement.

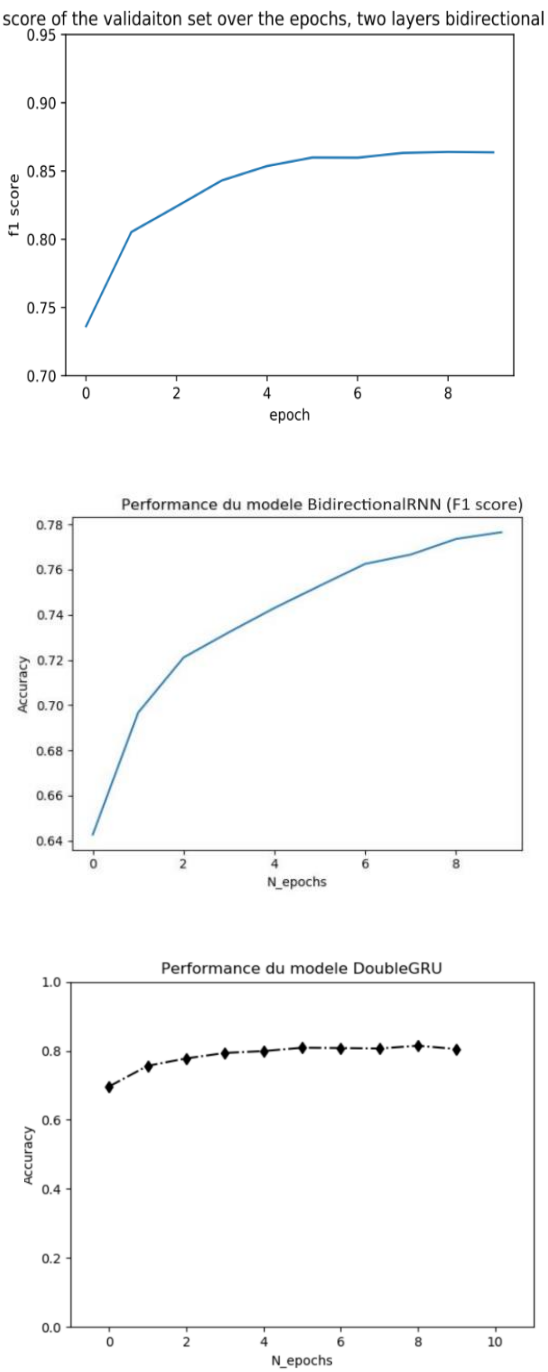

Figure 3: F1 score du jeu de validation en fonction du nombre d'épisodes pour différentes architectures

Tableau 4 : Temps d'entraînement des modèles

Architecture	Temps
LSTM simple	50 min 32 s
LTSM à deux couches	1 h 20 min 49 s
LSTM bidirectionnel	1 h 34 min 5 s
LSTM à deux couches bidirectionnel	1 h 40 min 48 s
GRU simple	46 min 14 s
GRU à deux couches	1 h 14 min 13 s
GRU bidirectionnel	1 h 34 min 5 s

Les GRU semblent s'entraîner un peu plus rapidement. Le GRU simple est 4% plus rapide que le LSTM simple. On trouve le même écart avec les architectures à deux couches. Les réseaux bidirectionnels ont eux le même temps d'entrainement.

Pour pouvoir conclure avec certitude sur la vitesse d'entraînement des modèles il aurait fallu utiliser des ensembles de données beaucoup plus grand. Un temps d'entraînement plus long aurait permis de faire apparaitre des plus gros écarts entre les architectures.

Nos observations semblent cependant logiques : les GRU ont une architecture moins complexe que les LSTM, ils semblent donc apprendre plus rapidement.

Discussion

Comparaison avec l'existant

Les scores obtenus dans la littérature avec des architectures similaire sont du même ordre de gradeur que les notre, c'est-à-dire un f1 score autour de 80% pour les meilleurs résultats.

La plus grande différence entre nos résultats et ceux déjà existants est l'écart de performance entre nos réseaux à GRU et LSTM qui est presque toujours supérieur à 5% alors qu'il est similaire dans les articles de référence. Une telle différence peut avoir deux explications. La première est la petite taille de nos jeux de données. Nous avons été limités par la puissance des machines à notre disposition. Plus de données aurait permis un meilleur apprentissage des modèles et aurait peut-être réduit l'écart. La deuxième raison réside peut-être dans les coefficients et fonctions d'activations utilisés au sein de réseau. Il est évident qu'elles ont une influence sur le traitement des données et donc les prédictions. Il serait pertinent d'entrainer d'autres modèles avec une même architecture mais des paramètres différents pour faire la comparaison.

Les meilleurs résultats de POS-Tagging de texte avec les réseaux neuronaux sont de près de 98% pour l'accuracy f1. Ils sont obtenus avec des réseaux LSTM-CRF, c'est-à-dire un LSTM couplé à des champs aléatoires conditionnels, permettant de mieux prendre en compte la contribution des différents nos d'une phrase.

Dans notre cas, on retrouve globalement de bons résultats avec les réseaux LSTM bidirectionnels qui performent bien mieux que les autres types de réseaux. Nous trouvons cependant des résultats qui différent des articles que nous citons en ce qui concerne les GRU. Dans nos expériences, ces derniers sont plus intéressants lorsqu'ils comportent deux couches cachées.

Limites de notre approche

Nous sommes restés dans l'étude d'un corpus de phrases anglaise. Le POS-Tagging est une des premières étapes de la traduction qui inclut au moins deux langues. Nous aurions donc aimé tester notre modèle sur des phrases autres que des phrases anglaises.

Nous aurions pu tester nos modèles sur des ensembles plus grands et sur plus d'ensembles, cependant, les contraintes ont été le manque de temps, puisque chacun des modèles met environ une heure à s'entraîner, voire plus dans le cas des modèles de type LTSM. Aussi, nous avons été limités par la performance de nos ordinateurs.

Conclusion

A travers les expériences menées, nous nous sommes rendu compte de l'utilité d'une cellule de mémoire dans les réseaux récurrents. Les RNN sont en effet bien moins performant que les LSTM ou les GRU. Contrairement aux résultats obtenus dans la littérature, nous avons remarqué une différence de performance entre les LSTM et les GRU. Pour conclure sur un tel écart, il faudrait cependant réaliser des expériences avec différents jeux de données dans plusieurs langues et de taille beaucoup plus importante.

Références

Juan Antonio Pérez-Ortiz, Mikel L. Forcada, *Part-of-Speech Tagging with Recurrent Neural Networks*, Departament de e Llenguatges i Sistemes Informatics, Universitat d'Alacant, E-03071 Alacant, Spain

Ashish Vaswani, Yonatan Bisk, University of Southern California, Kenji Sagae, Kitt.ai, Ryan Musa, University of Illinois at Urbana-Champaign, *Supertagging with LSTMs*

Raj Nath Patel, Prakash B. Pimpale, Sasikumar M., KBCS, CDAC Mumbai, *Recurrent Neural Network based Part-of-Speech tagger for Code-Mixed*, Nov 2016.

Xuezhe Ma and Eduard Hovy, *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*, Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Zhiheng Huang, Wei Xu, Kai Yu, *Bidirectional LSTM-CRF Models for Sequence Tagging*, Baidu research

M. Schuster and K. K. Paliwal, *Bidirectional recurrent neural networks*, in IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681, Nov 1997.

Ian H.Witten, Eibe Frank, Mark A. Hall, Christopher J. Pal, Data Mining (Fourth Edition) *Pratical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2017