Modèles heuristiques de classification des opinions à partir d'une étude comparative des réseaux profonds

Lisa Medrouk, Anna Pappa

Modèles heuristiques de classification des opinions à partir d'une étude comparative des réseaux profonds

Lisa Medrouk[1] et Anna Pappa[1]

[1]Université Paris 8, LIASD

9 juin 2017

Résumé

Dans cet article nous présentons une étude comparative de deux types d'architecture de réseaux profonds (Réseaux convolutionnels et RNNs, plus particulièrement, LSTM). Ensuite, nous proposons un modèle qui combine ces deux architectures : celle de CNNs qui sont hiérarchiques et sont adaptés à la détection des corrélations de voisinage, et celle de RNNs qui sont séquentiels et adaptés aux informations contextuelles à longue distance. Les CNNs et RNNs fournissent des informations complémentaires en classification de texte[VAGS16]. Nous souhaitons étudier cette complémentarité en l'appliquant à la problématique de classification d'opinions et de sentiments, et plus particulièrement à la détection de la polarité. Le corpus utilisé est en français mais les modèles sont également testés sur les données ouvertes IMDB [MDP+11]. Les résultats sont satisfaisants et l'expérience montre que le choix d'hyperparamètres est plus important que la profondeur des couches.

Mots-clef : Réseaux profonds, Analyse de sentiments, CNNs, RNNs, LSTMs.

1 Introduction

Le domaine de la fouille d'opinion et de l'analyse des sentiments a pris une grande place avec l'arrivée du Web communautaire et la multiplication des forums sur la toile. Le terme *Opinion Mining* est apparu pour la première fois dans un article de Dave et al. [DLP03] mentionnant le besoin de nouveaux algorithmes d'extraction et classification sémantique sur le retour d'opinions. D'autres termes y sont souvent associés, tels que l'analyse de l'affect, des émotions, de la subjectivité, l'extraction d'opinions, fouille de sentiments ou même de *review mining* [Liu12].

Notre étude porte sur la détection automatique de la polarité d'un sentiment en limitant les étiquettes possibles à deux catégories : les positives et les négatives. Les approches classiques, étudiées et présentées dans [PL08] sont à base de n-grammes et ne peuvent identifier avec précision les expressions complexes de sentiments. Ceci est dû à la perte d'information induite par l'approche «sac de mots» utilisée pour représenter les textes. Nous proposons d'utiliser des techniques d'apprentissage de réseaux de neurones profonds pour l'apprentissage des caractéristiques utiles à la détermination de la polarité d'une opinion.

En effet, les algorithmes d'apprentissage profonds ont récemment montré leurs performances dans les applications de traitement du langage naturel, y compris l'analyse du sentiment, et ce sur de multiples ensembles de données [CWB+11] [Kim14]. Un des grands avantages de ces modèles réside en leur capacité à extraire les caractéristiques d'apprentissage sans avoir à les leur fournir préalablement par un travail de pré-traitement. Les mots sont représentés par des vecteurs à hautes dimensions (*high dimensional vector space*), et c'est le modèle qui en extrait les caractéristiques utiles à son apprentissage.

Il existe deux grandes familles de réseaux profonds, les réseaux convolutionnels (CNN ou ConvNet) [LB98] et les Réseaux récurrents (RNNs) d'Elman [Elm90]. D'un point de vue architecture, les CNNs sont hiérarchiques, et les RNNs séquentiels. Les réseaux de neurones récurrents RNNs (*Recurrent Neural Networks*) sont particulièrement adaptés aux applications faisant intervenir le contexte. Des nouvelles variantes de RNNs ont été utilisées efficacement pour l'étiquetage de séquences [DI16]. Ils possèdent des connexions récurrentes qui permettent de prendre en compte à un instant t un certain nombre d'états passés. On parle alors de «mémoire

à court-terme≫. Néanmoins, pour s'assurer de la prise en considération de dépendances long terme nous avons choisi de travailler avec un type particulier des RNNs celui des LSTMs.

La première utilisation d'un réseau CNN en TALN a été réalisée par Collobert et Weston [CW08]. Plus récemment, les CNNs ont été utilisés pour la classification de phrases par [Kim14] qui a démontré l'efficacité d'une architecture peu profonde avec une seule couche de convolution suivie d'un *max pooling*. [dSdCG14] ont utilisé un modèle avec deux couches de convolution pour une extraction des caractéristiques pertinentes à partir de mots et de phrases pour l'analyse de sentiments de textes courts. [SM15] ont comparé différents modèles (Naïve Bayes, RNN, CNN, CNN + word2vec) sur le corpus de films Stanford *Sentiment Treebank* [SPW+13] obtenant la meilleure précision avec le modèle CNN + word2vec. [WLS+15] ont utilisé des LSTMs pour prédire la polarité de tweets. [KIJ+16] ont comparé les deux architectures (CNNs et RNNs) pour l'analyse de sentiments à partir des tweets en Russe. En architecture CNN profonde, [KGB14] ont proposé un modèle à cinq couches de convolution et ont introduit la notion de *temporal k-max-pooling* qui permet de détecter les caractéristiques les plus importantes d'une phrase.

Les CNNs et RNNs fournissent des informations complémentaires en classification de texte [VAGS16]. Les CNNs sont capables d'extraire les informations locales mais semblent échouer sur les dépendances longue distance ; les LSTMs permettent d'y remédier. Partant de ce constat, nous proposons de combiner ces deux architectures comme cela a été proposé par [ZSLL15] avec leur modèle C-LSTM, en adaptant les hyperparamètres et le nombre de couches, et ce afin d'étudier la complémentarité de ces deux architectures. Nous proposons de comparer différents modèles (LSTM, CNN, combinaison CNN LSTM, combinaison de deux couches convolutives et une LSTM) appliqués à un corpus de revues d'opinions en français, afin de démontrer la pertinence de la combinaison d'architectures profondes dans le cas d'une problématique de classification de la polarité de sentiments.

2 Réseaux LSTMs

Les réseaux de neurones récurrents RNNs (*Recurrent Neural Networks*) sont particulièrement adaptés aux applications faisant intervenir le contexte. Ils sont capables de mémoriser le passé proche, et commencent à *oublier* au bout d'une cinquantaine d'itérations. Ce phénomène, appelé dans la littérature *Vanishing Gradient Problem* correspond à une décroissance exponentielle de l'erreur après un certain nombre d'itérations. Ceci a été mis en évidence par Hochreiter et al. dans [HS97], qui pour remédier à ce problème ont mis au point des neurones particuliers, à mémoire large, les LSTMs (*Long Short-Term Memory*).

L'architecture LSTM est composée d'un ensemble de sous-réseaux récurrents particuliers situés au niveau de la couche cachée, et chacun contient une ou plusieurs *cellules de mémoire*. La première idée clé architecturale des réseaux LSTM est l'introduction d'un noeud spécial appelé CEC *Constant Error Carousel*. Il possède une connexion auto-récurrente avec un poids constant qui permet de collecter et de conserver les informations jugées pertinentes puis de les présenter au reste du réseau. Des portes multiplicatives qui sont des fonctions d'activation situées au niveau de l'entrée *input gate* et de la sortie *output gate* permettent de protéger respectivement l'état actuel de la mémoire et celui du reste du réseau (voir figure 1). Gers et al. [GSS03] ont proposé l'introduction d'une porte multiplicative supplémentaire appelée *porte d'oubli-forget gate* afin de permettre à la cellule de remettre à zéro le contenu du neurone et donc de réinitialiser l'état de la mémoire au cours de la séquence. Ainsi le CEC peut mémoriser les activations utiles tant que la porte d'oubli est ouverte, et de les oublier lorsqu'elle est fermée.

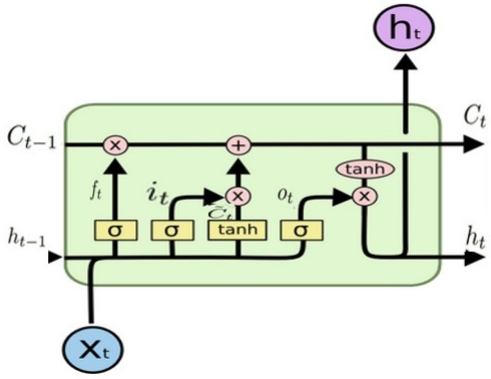

FIGURE 1 – Block LSTM proposé par [Ola15]

La figure 1 reflète un block LSTM déroulé, l'entrée h_{t-1} représente la sortie provenant du block précédent, C_{t-1}

représente la mémoire provenant du block précédent, X_t est le vecteur d'entrée.
h_t représente la sortie block courant, C_t représente la mémoire du block courant.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\widetilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$c_t = f_t \odot C_{t-1} + i_t \odot \widetilde{c}_t$$

$$o_t = \sigma(W_{io} \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \tanh(C_t)$$

Les $W_j \in R^{n \times m}$ sont des matrices de poids, les $b_j \in R^n$ sont des vecteurs de biais, pour $j \in \{i, f, c, o\}$. $\sigma, \tanh$ sont respectivement les fonctions sigmoïde et tanh. $\odot$ est une multiplication point par point. f_t correspond à la porte d'oubli, c_t la mise à jour des états, o_t, h_t les sorties, $\widetilde{c}_t, i_t$ les nouvelles informations à conserver.

3 Réseaux de neurones convolutionnels

Les réseaux de neurones convolutionnels ont initialement été inspirés par la découverte faite par Hubel et Wiesel [DT62] de neurones sensibles aux aspects locaux et sélectifs en orientation, dans le système visuel du chat. Les réseaux de neurones convolutifs sont basés sur le perceptron multicouches. Ce sont des réseaux de neurones acycliques (*feed-forward*) adaptés à la détection des corrélations de voisinage, ce qui les rend particulièrement intéressants aux traitements appliqués au langage naturel. Les CNNs ne nécessitent qu'un très faible taux de pré-traitement nécessaires à leur fonctionnement, et ne requièrent aucun choix d'extracteur de caractéristiques spécifiques. En traitement d'image, les modules dits de convolution prennent une image X de taille $l_i * h_i$ en entrée et la convoluent par un filtre de poids K de taille $l_k * h_k$. Le masque de convolution balaye toutes les positions sur l'image d'entrée. Dans le cas d'un texte, le masque de convolution est appliqué à tous les n-grammes du texte en entrée, chaque couche de convolution applique le même traitement local à l'ensemble de la couche précédente. Ce traitement dit *noyau de convolution* fait ressortir certaines caractéristiques locales [LB98].
Soit un filtre représenté par la matrice W_f ; Une convolution sur n vecteurs des mots consécutifs à partir de la position i se décline comme suit :

$$c_i = ReLu(W_f \cdot X_{i:i+n-1} + b_f)$$

$(\cdot)$ étant le produit matriciel, b étant le biais. ReLu étant une fonction non-linéaire. L'apprentissage est basé sur une rétro-propagation classique stochastique.

4 Corpus utilisé

Nous avons constitué un corpus en langue française, extrait d'opinions liées à des restaurants. Nous avons relevé 65242 opinions notées de 1 à 5 (1 étant la note la plus basse), afin de minimiser l'effet subjectif des phrases, nous avons choisi d'éliminer les opinions notées 3, ce qui constitue un corpus de travail composé de 47818 opinions (23909 positives et 23909 négatives). La taille moyenne des séquences est de 429 caractères. Les couches de convolutions exigent des séquences à taille fixe, nous les avons établies à 500, par la technique de *pad*, les séquences les plus longues sont coupées et les plus courtes sont complétées par des 0. Le corpus n'a subi aucun autre pré-traitement. Les modèles sont testés une première fois avec le corpus brut vectorisé, puis par des vecteurs-mots de taille 50 et 300 appris sur la Wikipédia française avec l'outil word2vec [MCCD13]. Le *Word embeddings* est une approche de la sémantique distributionelle qui permet de représenter des mots sous la forme de vecteurs de nombres réels.
Afin de pouvoir comparer la performance de notre modèle nous avons travaillé avec les données de classification de sentiments IMDB [MDP+11], constituées de 100 000 revues de films (25 000 positives, 25 000 négatives et 50 000 non annotées). La taille moyenne des séquences étant de 231 mots.
En fonction de la présence de certains marqueurs dans l'énoncé, la polarité peut être implicite ou bien explicite. En utilisant notre configuration du modèle superposé CNN-LSTM, nous nous intéressons principalement à la notation "positive" ou "négative" intrinsèque des mots, et aux mots dont l'orientation peut changer selon le contexte, ainsi qu'aux inversions du sens par la distance des morphèmes discontinus (négation).

5 Nos Modèles

Notre premier modèle est un réseau convolutionnel peu profond inspiré par Kim [Kim14], le second est un réseau LSTM simple, le troisième est une combinaison de deux précédents, et enfin le quatrième est composé

de deux couches de convolutions suivi d'une couche LSTM. Ce dernier modèle composé de dix couches au total a l'architecture la plus profonde et se décline comme suit :

— La Couche d'entrée, c'est l'espace sémantique représenté par la taille des word embedding (15266 pour les modèles pré-entraînés et 5000 pour le corpus brut), dans un espace de projection de taille 34,50,300 pour respectivement le corpus brut, word2vec 50 et word2vec 300 le tout pour une taille de séquence fixe à 500.

— Une couche de régularisation de type Dropout à 0.25 [1]. Ce type de régularisation a été introduit par [KSH12]. Le dropout permet de masquer un sous-ensemble aléatoire d'unités cachées vues lors de l'optimisation, ce qui les empêche de fonctionner en groupe, leur permettant ainsi de devenir plus indépendantes. De fait, nous entraînons un grand nombre de réseaux différents, partageant tous partiellement le même poids. Cela nous assure qu'aucune unité particulière ne soit trop importante dans la prise de décision.

— Une couche convolutionnelle avec 64 filtres, un filtre de convolution de taille 5 et une activation de type *ReLU Rectified Linear Unit* pour unité de rectification linéaire, $F(x) = max(x, 0)$ égal à 1 si le signal en entrée est positif, ou 0 s'il est négatif. La ReLu augmente les propriétés non linéaires de la fonction de décision et de l'ensemble du réseau sans affecter les champs récepteurs de la couche de convolution.

— Une couche d'agrégation dit *pooling* permettant principalement de détecter des motifs récurrents dans une phrase [KGB14]. Un pooling est une forme de sous-échantillonnage qui permet de compresser l'information en réduisant sa taille, permettant ainsi de réduire le nombre de paramètres à calculer et de contrôler le surapprentissage. Nous avons choisi de travailler avec un pooling de type *max* qui ne conserve que la plus grande valeur de la taille de la sous-région sélectionnée (4 dans notre cas).

— Une deuxième couche convolutionnelle avec 64 filtres, un filtre de convolution de taille 5 et une activation de type *ReLU*.

— Une couche agrégation-max (max-pooling) avec taille de sous-région à 4

— Une couche de régularisation de type Dropout à 0.4.

— Une couche LSTM à 70

1. voir section Hyperparamètres et apprentissage

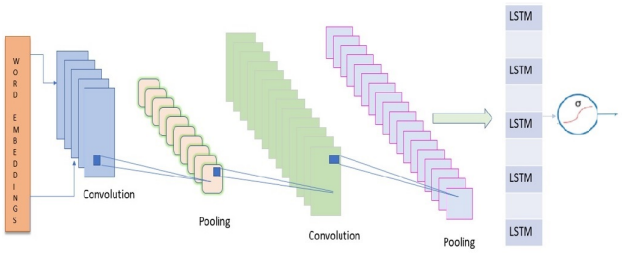

FIGURE 2 – Architecture profonde proposée

— Une couche totalement connectée dite "FC fully connected" à 1 correspondant au nombre de classe attendu.

— La dernière couche est une fonction d'activation sigmoïde $f(x) = (1 + e^{-x})^{-1} f : R \rightarrow [0, 1]$ permettant d'obtenir des probabilités d'appartenance à chaque classe.

L'apprentissage est réalisé par la méthode de descente de gradient stochastique, optimisée par la technique Adam [KB14] qui calcule un taux d'apprentissage pour chaque paramètre.

5.1 Hyperparamètres et apprentissage

Entraîner un modèle implique généralement de choisir la famille du modèle puis de choisir les hyperparamètres correspondants. Pour notre protocole de tests, nous avons choisi d'avoir les mêmes hyperparamètres sur les différentes architectures afin d'établir notre comparatif de modèles. Le nombre de tests pour arrêter le choix de ces hyperparamètres a été réalisé par un processus heuristique manuel (voir tableau 1). Nous avons expérimenté les différentes valeurs possibles de chaque hyperparamètre sur les différentes architectures afin de ne retenir que celles qui permettaient de retourner les meilleures performances ($> 85\%$) tout en ayant une bonne capacité de généralisation. Les tests ont été mené sur une base de 8 itérations pour chaque modèle. Par exemple, pour le filtre linéaire des couches de convolutions, nous avons testé les différentes architectures avec un filtre de taille 3, puis 4, et ainsi de suite jusqu'à 7, pour choisir finalement le filtre de taille 5 avec lequel la performance de différents modèles était toujours supérieure à 85% avec une capacité de généralisation du même ordre.

Nous avons réalisé nos tests avec la librairie open source Keras [Cho15]. Le *deep learning* faisant un usage intensif des calculs matriciels, nous avons travaillé par conséquent avec les unités de traitement

graphique (GPUs) NVIDIA 940M.

Hyperparamètres	Tests	Choix
Nbr de filtres CNN	30-100	64
filtres linéaires	3-7	5
Régularisation-Dropout	0-1	0.25 puis 0.4
Agrégation-Pool	2-4	4
Taille LSTM	50-200	70
Optimisation	Adam, RMSprop	Adam

Table 1 – Sélection des hyper-paramètres

6 Résultats

Nous présentons en premier lieu, l'évaluation de notre modèle sur les données ouvertes IMDB [MDP+11], et ce, afin de pouvoir les comparer à l'état de l'art. Nous présentons les résultats sur les données IMDB brutes et pré-entraînées par des vecteurs-mots GloVe [PSM14]. GloVe est un modèle pré-entraîné sur les phrases et mots de Wikipédia pour le prolongement de mots. Chaque mot est représenté par un vecteur de longueur 300 et représente la corrélation entre les mots. Nous initialisons nos vecteurs aux poids correspondants à ceux de GloVe.

Modèles	Performance
LSTM[HF15]	89.1
Model Unlabeled + Bag of Words (bnc)[MDP+11]	88.89
LSTM	87.06
LSTM - Glove	89.57
CNN	89.16
CNN - Glove	89.54
CNN-LSTM	89.18
CNN-LSTM - Glove	89.90
BI CNN-LSTM	89.48
BI CNN-LSTM - Glove	90.34

Table 2 – Résultats obtenus pour les données IMDB

Les résultats du modèle BI CNN-LSTM pré-entraîné avec Glove sont les plus performants. L'amélioration est légère avec un delta de 0,5.

Le troisième tableau relève la performance moyenne obtenue en entraînant les différents modèles sur notre corpus français brut ou pré-entraîné par des vecteurs-mots word2vec de taille 50 et 300. L'initialisation des poids étant aléatoire et l'addition de la technique du dropout introduisant également en effet aléatoire dans le choix des données entraînées, impliquant des résultats stochastiques, nous avons donc effectué plusieurs runs afin de proposer une moyenne pour la performance. Les modèles sont entraînés sur 33473 opinions et validés sur 14345 opinions. Les résultats démontrent une bonne performance pour toutes les architectures proposées. Un très léger mieux est à noter pour l'architecture BI CNN LSTM sur des données brutes (un delta de $0,18\%$) en comparaison avec une architecture de type CNN simple. Le temps d'exécution du modèle CNN simple étant de 3,6 minutes, et de 14,33 minutes pour un modèle BI CNN LSTM et allant jusqu'à 25,3 minutes pour un BI CNN LSTM 300.

Modèles	Performance
BI CNNs LSTM 300	94.86
BI CNNs LSTM 50	93.08
BI CNNs LSTM	94.90
CNN LSTM 300	94.41
CNN LSTM 50	93.43
CNN LSTM	94.74
LSTM 300	94.03
LSTM 50	92.80
LSTM	94.33
CNN 300	93.96
CNN 50	92.76
CNN	94.71

Table 3 – Résultats corpus français pour une classification binaire

Le quatrième et cinquième tableau sont des rapports de classification des modèles CNN 300 et BI CNN 300, pour 14345 données de validation en langue française (7156 positives et 7189 négatives), les résultats en Précision, Rappel et f1-score permettent de mieux préciser la robustesse du modèle. Exemples de revues mal qualifiées avec un modèle CNN (relevées sans correction) :

BI CNN LSTM	**Précision**	**Rappel**	**F1-**mesure	**Nbr d'**opinions **de test**
Négatif	0.93	0.95	0.94	7189
Positif	0.95	0.93	0.94	7156
	0.94	0.94	0.94	14345

Table 4 – Rapport de classification : Précision, Rappel et F1-score pour le Modèle CNN 300WV

	Opinions Positives	Opinions Négatives
Opinions Positives	6853	336
Opinions Négatives	521	6635

TABLE 5 – Matrice de confusion Modèle CNN 300 pour 14345 revues

"bonne ambiance, bonne adresse, belle clientele, mais nourriture tres moyenne. un endroit que l'on frequente que pour l'ambiance pas pour le contenu de l'assiette...j'ai joué le jeu un soir, je n'y retournerai pas...mes amis non plus. je ne m'attendais pas à un gastro mais compte tenu des prix, je pensais que cela serait meilleur."
Cette revue a été annotée en "Négative" et a été prédit en Positive.
"emplacement au top, à quelques pas de l'arc de triomphe, cuisine goûteuse et recherchée, on sent la maîtrise en cuisine, proportions correctes cependant, ce resto a des points à améliorer pour prétendre être gastro et digne de ses promesses. la déco n'est pas raffiné, l'ambiance n'est pas intimiste, je vous en supplie, arrêter le set de table en papier !...".
Cette revue a été annotée en "Positive" et a été prédit en Négative.

BI CNN LSTM	**Précision**	**Rappel**	**F1-**mesure	**Nbr d'** opinions **de test**
Négatif	0.96	0.94	0.95	7189
Positif	0.94	0.96	0.95	7156
	0.95	0.95	0.95	14345

TABLE 6 – Rapport de classification : Précision, Rappel et F1-score pour le Modèle BI CNN LSTM 300

	Opinions Positives	Opinions Négatives
Opinions Positives	6736	453
Opinions Négatives	298	6858

TABLE 7 – Matrice de confusion Modèle BI CNN LSTM 300 pour 14345 revues

Exemples de revues mal qualifiées avec un Modèle composé d'un LSTM :
"bonne ambiance, bonne adresse, belle clientele, mais nourriture tres moyenne. un endroit que l'on frequente que pour l'ambiance pas pour le contenu de l'assiette...j'ai joué le jeu un soir, je n'y retournerai pas...mes amis non plus. je ne m'attendais pas à un gastro mais compte tenu des prix, je pensais que cela serait meilleur."
Cette revue a été annotée en "Négative" et a été prédit en Positive.
"un établissement que je connais depuis mon enfance, offrant une carte intéressante avec des très bons plats. le personnel n'est pas souvent à la hauteur du lieu : certaines serveuses sont à la limite du désagréable... heureusement, nous avons notre serveur préféré (des îles...). il est là pour "relever le niveau".
Cette revue a été annotée en "Positive" et a été prédit en Négative.
Nous relevons dans le premier exemple de 'phrase mal qualifiée' que le modèle LSTM et CNN ont traité cette revue de la même façon. Dans ce cas, le modèle LSTM n'a pas mieux géré les dépendances longue distance qu'un modèle CNN. Un même constat est à porter pour d'autres résultats de type faux positifs et faux négatifs obtenus.
Les résultats obtenus sont satisfaisants avec une forte performance globale. Nous notons une très légère amélioration avec une architecture BI CNN LSTM, mais qui implique un temps d'entraînement bien plus long. Ce faible delta relevé par l'architecture Bi CNN LSTM ne permet pas de prouver la complémentarité des deux approches. Nous relevons également que les Modèles composés d'une couche LSTM n'ont pas mieux traité les dépendances longues distances pour les revues de test qu'un balayage de taille 5 n-grammes des modèles CNN proposés. En prenant en compte les performances relevées et le temps d'exécution, nous permet de choisir une architecture CNN pour ce type de classification. Nous relevons également de bonnes performances pour les modèles exécutés sur du texte brut sans pré-entraînement peut être dû à la nature des revues relevées sur le web contenant les inévitables fautes de frappe, fautes d'orthographe, ..etc.

7 Conclusion

Nous avons réalisé un travail comparatif de modèles de réseaux neuronaux peu profonds combinant deux types d'architectures. Les modèles ont été testé dans le cadre de classification de la polarité de revues d'opinions à l'échelle des phrases, prenant en entrée des vecteurs-mots appris selon la méthode Word2vec ainsi que sur du texte brut (non structuré). Nous avons montré que les performances obtenues sont au niveau des meilleurs algorithmes dans le domaine, ce qui nous

conforte dans notre choix des réseaux profonds au niveau classification de texte. Nous avons également relevé que la combinaison des différentes architectures n'apportait pas une amélioration conséquente en termes de performance. Notre comparaison montre également que les modèles LSTM ne capturent pas mieux les dépendances longue distance que le modèle CNN avec un filtre de taille bien choisi. Notre expérience montre que le choix des hyperparamètres semble plus important que la profondeur des couches, ou la combinaison choisie. Afin de pousser nos recherches nous envisageons de renouveler l'expérience sur un grain plus fin tels que les caractères.

Références

[Cho15] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[CW08] Ronan Collobert and Jason Weston. A unified architecture for natural language processing : Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167, New York, NY, USA, 2008. ACM.

[CWB+11] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12, 2011.

[DI16] Marco Dinarelli and Tellier Isabelle. Étude de réseaux de neurones récurrents pour l'étiquetage de séquences. In *JEP-TALN-RECITAL 2016, volume 2 : TALN*, pages 98–111, Paris, France, 2016. Association pour le Traitement Automatique des Langues.

[DLP03] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery : Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 519–528, New York, NY, USA, 2003. ACM.

[dSdCG14] Cícero Nogueira dos Santos and Maíra A. de C. Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, 2014.

[DT62] Hubel DH and Wiesel TN. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 1962.

[Elm90] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2) :179–211, 1990.

[GSS03] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *J. Mach. Learn. Res.*, 3 :115–143, March 2003.

[HF15] James Hong and Michael Fang. Sentiment analysis with deeply learned distributed representations of variable length texts. 2015.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8) :1735–1780, November 1997.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[KGB14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *CoRR*, 2014.

[KIJ+16] Arkhipenko K., Kozlov I., Trofimovich J., Gomzin A., and Turdakov D. Skorniakov K. Comparison of neural network architectures for sentiment analysis of russian tweets. *Computational Linguistics and Intellectual Technologies : Proceedings of the International Conference "Dialogue 2016"*, 2016.

[Kim14] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.

[LB98] Yann LeCun and Yoshua Bengio. The handbook of brain theory and neural networks. chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. MIT Press, Cambridge, MA, USA, 1998.

[Liu12] Bing Liu. *Sentiment Analysis and Opinion Mining*. Morgan and Claypool, 2012.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[MDP+11] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies - Volume 1*, HLT '11, pages 142–150, 2011.

[Ola15] Christopher Olah. Understanding lstm networks, 2015.

[PL08] Bo Pang and Lillian Lee. Opinion mining et sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2) :1–135, January 2008.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove : Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[SM15] Houshmand Shirani-Mehr. Applications of deep learning to sentiment analysis of movie reviews. 2015.

[SPW+13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics, 2013.

[VAGS16] Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. Combining recurrent and convolutional neural networks for relation classification. *CoRR*, abs/1605.07333, 2016.

[WLS+15] Xin Wang, Yuanchao Liu, Chengjie Sun, Baoxun Wang, and Xiaolong Wang. Predicting polarities of tweets by composing word embeddings with long short-term memory. In *ACL*, 2015.

[ZSLL15] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Chi-Moon Lau. A c-lstm neural network for text classification. *CoRR*, abs/1511.08630, 2015.