Évaluation de la positivité d'un tweet à l'aide des LSTM

Cédric Grelier and **Lorent Caravaku**
Polytechnique Montréal
Montréal, Canada
cedric.grelier@polymtl.ca
lorent.caravaku@polymtl.ca

Abstract

Cet article a pour objectif de présenter notre travail sur l'évaluation de la positivité d'un tweet à l'aide des LSTM. Nous avons évalué différents modèles et ainsi comparé les résultats obtenus avec ceux de l'état de l'art. A travers l'article, nous présenterons les données utilisées, les modèles implémentés et nous discuterons enfin des résultats.

Introduction

Nous avions découvert la compétition SemEval dont l'un des objectifs était l'évaluation de la positivité d'un tweet. Plus particulièrement, la base de notre recherche fut le papier de conclusion de l'année précédente (Rosenthal, Farra, and Nakov 2017). Sur ce sujet, de nombreux articles ont été publiés, nous en reparlerons dans la partie suivante. En plus de ce point, il nous était important d'utiliser un LSTM (Hochreiter and Schmidhuber 1997) en rapport avec notre article de présentation (Chung et al. 2014). C'est pourquoi nous avons décidé d'utiliser une architecture ressemblant à ce que l'on peut trouver ces dernières années dans la littérature, notamment dans la compétition SemEval, afin d'adresser la question de la classification de la positivité d'un tweet. Dans ce contexte, nous avons réussi à obtenir des résultats proches de l'état de l'art actuel tout en étant restreints par des limites temporelles, matérielles et financières.

Travaux connexes

L'analyse automatique de texte a suscité de nombreux travaux depuis les années 2000. Les intérêts pour mener de tels travaux n'ont fait qu'augmenter avec l'arrivé des données massives générées par les réseaux sociaux, comme Twitter et Facebook.

Dans les années 2000 deux grands types de méthode étaient utilisés pour l'analyse de sentiments. La première méthode consistait en l'utilisation de données étiquetées pour entrainer des algorithmes d'apprentissage automatique avec des méthodes de classificaiton, comme la classification naïve bayésienne et les machines à vecteurs de support (Bo Pang 2002). La deuxième méthode consistait en l'utilisation d'un lexique de mots associés à des sentiments pour ainsi évaluer le sentiment d'un text à travers des fonctions de comptage spécifiques (Xiaowen Ding 2008).

Depuis, avec l'arrivé de nouveaux algorithmes d'apprentissage automatique, et plus particulièrement ceux concernant l'analyse de séries temporelles, comme le LSTM (Long Short-Term Memory), les travaux ont évolué et apporté de meilleurs résultats ces dernières années dans le cas du traitement de texte automatique (Duyu Tang 2015).

Durant ces 10 dernières années, le problème de l'analyse des sentiments est devenu de plus en plus populaire et diverses compétitions sont organisées chaque année. Des acteurs venant de milieux différents, académique et industriel, participent et les résultats ne font qu'évoluer. En guise d'exemple, SemEval (Semantic Evaluation) est une série d'évaluations de systèmes d'analyse sémantique, organisée tous les ans sous l'égide de SIGLEX (Special Interest Group on the Lexicon of the Association for Computational Linguistics). La tâche 4 de SemEval (Rosenthal, Farra, and Nakov 2017) a pour objectif l'analyse de sentiments sur des publications de Twitter, c'est sur quoi notre travail porte. Dans cette catégorie, les deux vainqueurs de l'édition 2017 de SemEval étaient : (Cliche 2017) et (Baziotis, Pelekis, and Doulkeridis 2017). Ces deux participants ont eu de bons résultats : en terme de précision, le premier a obtenu 88,2% et le deuxième 85,6%.

Limités par la mémoire disque disponible sur les machines du laboratoire, nous étions incapable de travailler avec les données fournie par SemEval. Par conséquent, notre travail a été effectué avec la base de données Sentiment140, contenant uniquement 1,5 million de Tweets. En 2009, des travaux avec cette base de données ont déjà été effectués par des chercheurs (Alec Go 2009). Employant des méthodes anciennces (SVM, classification bayésienne naïve), ils ont obtenu des résultats dépassant les 80% de précision. Nous comparerons nos résultats aux leurs.

Approche théorique

Données

Dans le domaine du traitement du langage naturel, de nombreux corpus de données sont disponibles. Toutefois, dans le cas du LSTM et des réseaux de neuronnes profonds, il est indispensable d'en avoir suffisamment pour pouvoir entraîner efficacement nos modèles. Malheureusement, le nombre de phrases ou tweets nécessaires est très élevé ce qui rend compliquée l'approche manuelle pour la construction des corpus.

De fait, (Go, Bhayani, and Huang 2009) propose une approche automatique se basant sur les émoticonnes. Le corpus utilisé est disponible à cette adresse `http://help.sentiment140.com/for-students/`. Il est découpé en deux corpus distinct d'entraînement et de test. Les labels utilisés classifient les tweets en 3 : *négatif*, *neutre* ou *positif*. Toutefois, il n'y a aucun tweet neutre dans le corpus d'entraînement, seulement dans le corpus de test. Cela est du à la manière utilisée pour créer le corpus, les émoticonnes ne permettant pas de classer automatiquement un tweet comme neutre.

C'est pourquoi nous avons décidé de ne travailler qu'avec les labels *négatif* et *positif*, car ce corpus est le plus complet disponible à notre connaissance. De plus, il aurait été long et fastidieux de créer et/ou compléter le corpus avec des tweets neutres étant donné le peu de temps dont nous disposions. Nous utilisons donc les deux corpus mais sans les exemples de tweet *neutres* dans le corpus de test.

Nettoyage des données Il est à noter que puisque les données sont issues de tweets en ligne, elles sont polluées par toute sorte de choses. Principalement, nous avons effectué un nettoyage sur les particularités suivantes :

— les hashtags (#exemple)
— les urls (http ://www.exemple.com)
— les liens des noms d'utilisateurs (@exemple)

En enlevant toutes les occurences dans les tweets pour ne garder que l'essentiel du message. L'objectif étant de ne pas obstruer l'apprentissage avec des données parasites. L'importance de ces éléments est non-négligeable au vu de la diminution en taille du fichier qui est passé de 233kO à 123kO. Peu de tweets ont été complètement supprimés (vides après nettoyage).

Voir Table 1 pour la taille des corpus utilisés.

	Entraînement	Test
Nombre d'éléments	1599961	359

TABLE 1 – Nombre d'éléments dans le corpus (nettoyé) Sentiment140

Prolongement de mots

Le prolongement de mots ou "word embedding" en anglais, est un ensemble de techniques permettant de pointer le sens sémantique dans un espace géométrique. Il est question d'associer à un mot une représentation vectorielle numérique. Cela facilite les tâches d'apprentissage automatique. Particulièrement dans notre cas, tous nos modèles seront précédés d'une couche (Embedding en Keras) permettant d'apprendre ces représentations au fur et à mesure. Il existe plusieurs méthode de prolongement de mots comme Word2Vec (Mikolov et al. 2013) ou encore GloVe(Pennington, Socher, and Manning 2014).

Pré-entraînement Il est toutefois possible d'utiliser des espaces déjà entraînés afin d'améliorer notre prolongement. De fait, nous utilisons pour certains modèles un fichier glove.twitter.27B.50d tiré de `https://nlp.stanford.edu/projects/glove/` contenant des mots représentés par des vecteurs de dimensionalité 50. Il a été entraîné sur 2B de tweets contenants 27B mots distincts. L'intérêt d'utiliser un tel fichier est d'augmenter notre précision sur la représentation sémantique des mots en ayant dès le début une représentation correcte.

Modèles

Nous avons élaboré 7 modèles d'apprentissage différents. Avec les résultats de chacun d'entre eux nous pourrons ainsi déterminer lequel performe le mieux. Outre la précision, nous pourrons aussi discuter du temps d'entraînement de chacun de ces modèles en fonction de leurs configurations.

Les modèles ont tous une voire plusieurs couches LSTM. Cette dernière est la plus importante couche pour la résolution de notre problème. Il est bon de rappeler que LSTM est ce qu'il y a de plus intéressant, à l'heure actuelle, pour l'analyse de séries temporelles, en l'occurrence, ici, du traitement de texte. La plupart des modèles utilisent un pré-entraînement, certains non. Des couches Dropout sont utilisées pour éviter le surentraînement et ainsi avoir de meilleurs résultats. Et, enfin, des couches de convolution sont appliquées au début de chacun des modèles, sauf pour le 5ème, ceci afin de voir en quoi la convolution peut impacter la précision et le temps d'entraînement.

Voir Table 2 pour la configuration des modèles.

Voir Table 3 pour le nombre de paramètres de chaque modèle.

Expériences

L'entraînement des différents modèles est effectué sur les données d'entraînement Sentiment140 présentées dans le paragraphe "Approche théorique". L'arrêt de l'entraînement est basé sur un $early\_stopping$ ayant pour $monitor$ la précision des données de validation et une $patience$ de 3. Les données de validation sont obtenues en divisant les données d'entraînement avec un facteur $validation\_split$ de 0.2.

Résulats

Sur les deux graphiques (Figure 1 et Figure 2) sont représentées la précision et la perte sur les données de

Modèle	Couches
1	Embedding (Pre-embedding) Conv1D-128 Conv1D-64 Conv1D-32 MaxPooling1D Dropout Bidirectional LSTM-128 Bidirectional LSTM-128 Dropout Dense
2	Embedding (Pre-embedding) Conv1D-256 MaxPooling1D Dropout LSTM-128 Dropout Dense
3	Embedding (NON Pre-embedding) Conv1D-64 MaxPooling1D Dropout Bidirectional LSTM-128 Dropout Dense
4	Embedding (Pre-embedding) Conv1D-64 MaxPooling1D Dropout Bidirectional LSTM-128 Dropout Dense
5	Embedding (Pre-embedding) Dropout Bidirectional LSTM-64 Bidirectional LSTM-64 Dropout Dense
6	Embedding (Pre-embedding) Dropout Conv1D-256 MaxPooling1D Dropout LSTM-128 Dropout Dense
7	Embedding (Pre-embedding) Dropout Conv1D-64 MaxPooling1D LSTM-70 Dense

TABLE 2 – Les différents modèles

validation en fonction du nombre d'epochs. Ce que nous constatons c'est que, dans l'ensemble, tous les modèles ont

Modèle	Nombre de paramètres
1	16,574,012
2	16,213,084
3	16,165,660
4	16,165,660
5	16,109,404
6	16,213,084
7	16,043,396

TABLE 3 – Nombre de paramètres

des résultats relativement proches. Toutes les courbes sont en forme de cloche, ceci s'explique par le sur-entraînement des modèles à partir d'un certain nombre d'epochs. Sur les données de validation, comme nous pouvons le voir sur les deux graphiques, le modèle qui donne les meilleurs résultats est le modèle 5 : la précision atteint une valeur de 83,8% et la perte une valeur de 36,5%. Cependant, sur les donnés de test ce n'est plus le cas, c'est plutôt le modèle 7 qui remporte la première place avec une précision de 82,7% (c.f. Table 4). La configuration du modèle 7 est présentée par la Figure 3.

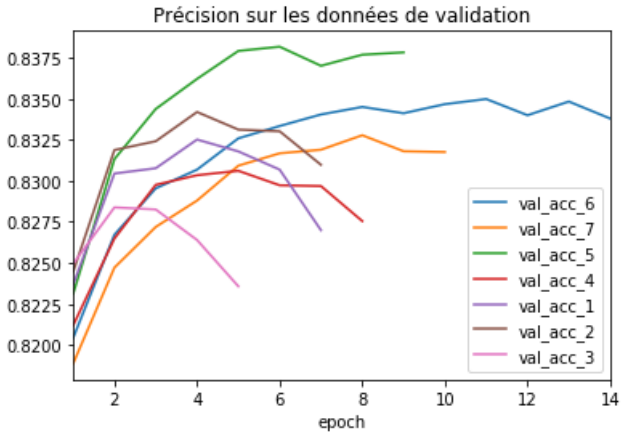

FIGURE 1 – Evolution de la précision sur les données de validation en fonction du nombre d'epoch

En ce qui concerne la durée des epochs, encore une fois les résultats restent relativement proches pour tous les modèles à part le 1er et le 5ème. En effet, contrairement aux autres modèles qui ont des epochs qui durent environ 1 500 secondes, les epochs du 1er modèle durent environ 3 000 secondes et ceux du 5ème modèle environ 10 200.

Les analyses concernant ces résultats seront présenté dans le paragraphe suivant.

Analyse

Après ces expériences, et plus particulièrement les résultats, dans ce qui suit, nous tenterons d'expliquer les conclu-

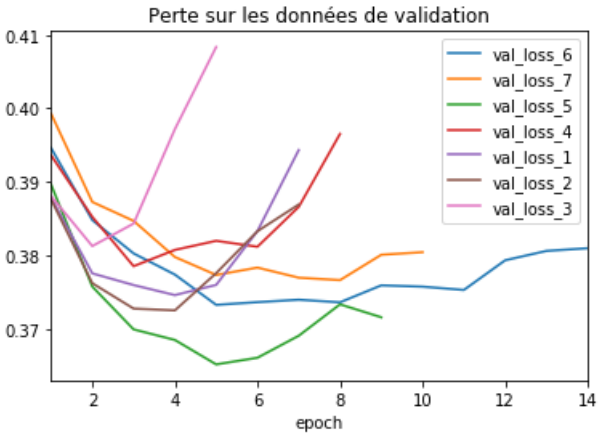

FIGURE 2 – Evolution de la perte sur les données de validation en fonction du nombre d'epoch

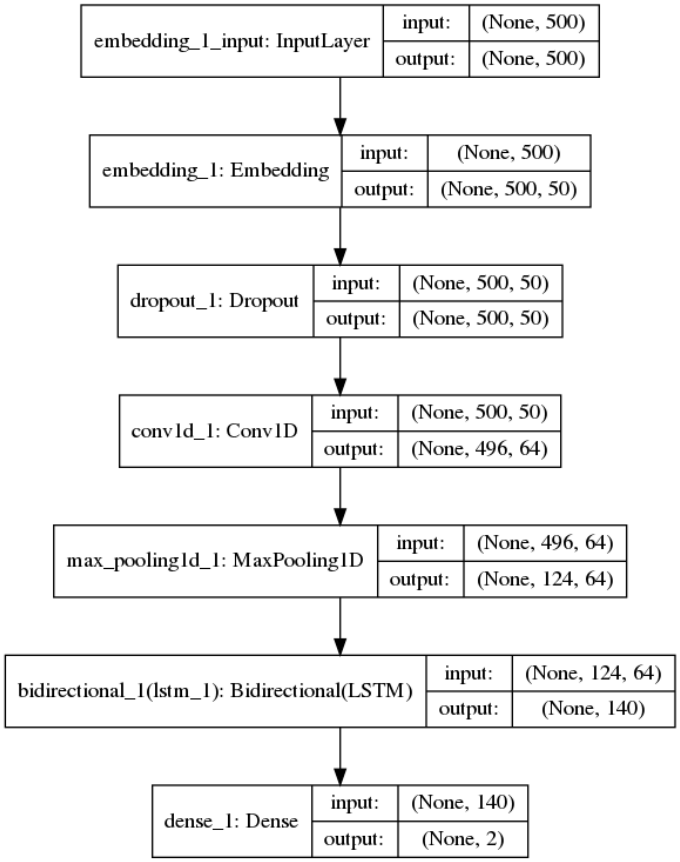

FIGURE 3 – Configuration du modèle 7

Modèle	Précision	Perte	Temps par epoch (s)
1	0.8050	0.4469	3 000
2	0.7966	0.4626	**1 390**
3	0.8161	0.4283	1 450
4	0.8105	0.4378	1 450
5	0.7994	0.4270	10 200
6	0.8161	0.3993	1 450
7	**0.8272**	**0.3954**	1 550

TABLE 4 – Précision et perte obtenues sur l'ensemble de test pour les différents modèles

sions que nous pouvons en tirer et les améliorations possibles pour de futurs expériences.

Discussion

Les sujets sur lesquels nous discuterons sont :
— l'impact de l'utilisation d'un pré-entraînement,
— l'utilisation d'un ensemble de données de test construit différement que le reste qui est probablement à l'origine des différences entre les résultats sur les données de validation et les résultats sur les données de test,
— l'impact de la configuration du modèle sur le temps de calcul par epoch, et finalement
— l'effet des hyperparamètres sur les résultats.

Pré-entraînement Nous avons expressément élaboré exactement deux modèles identiques qui ne différent que par le fait qu'un d'entre eux est avec pré-entraînement et l'autre non (modèle 3 et 4). Avec les résultats obtenus nous pourrons ainsi voir l'effet du pré-entraînement dans le cas de notre sujet.
Sur les données de validation, nous constatons que le modèle avec pré-entraînement (modèle 4, courbe rouge) obtient de meilleurs résultats et arrive au stade du surentraînement plus tard que le modèle sans pré-entraînement (modèle 3, courbe rose). Le temps de calcul par epoch reste le même pour les deux modèles. Nous pouvons ainsi conclure que l'utilisation de matrice pre-embedded ne peut être que positif pour l'apprentissage dans notre cas de figure.

Surentraînement/Données de test Les données de test n'ont pas été construites de la même manière que les données d'entraînement et de validation. L'ensemble d'entraînement a été construit de manière automatique à partir des tweets (avec émoticonnes) directement collectés sur le réseau social, les données ont été étiquetées en se basant sur les émoticonnes présents sur le tweet en question. Tandis que, l'ensemble de test a été construit manuellement à la main. Ce point de différence entre ces deux ensembles est à l'origine d'une non-homogénéité de ces deux derniers.
Nous sommes fort convaincus que la différence des résultats constatée dans le paragraphe "Résultats" est due à cette différence présentée précédemment.

Temps de calcul par epoch Pour chacun des modèles nous avons prélevé le temps de calcul par epoch pour ainsi déterminer ce qui peut entraîner une augmentation ou une

diminution de celui-ci.
Les résultats restent relativement proches pour tous les modèles, exceptés le 1er et le 5ème. Pour le premier modèle, le temps de calcul qui est 1 fois plus grand que celui des autres modèles, s'explique par les 3 couches de convolution successives. Une couche de convolution requiert des opérations qui ne sont pas négligeables en termes de temps de calcul. Il en est de même lorsque nous retirons la (les) couche(s) de convolution : si aucune couche de convolution est présente à l'entrée de notre modèle alors le nombre de variables à traiter par le LSTM est grand, par conséquent cela prend beaucoup de temps ; ce phénomène est observé avec le modèle 5 (10 200 secondes par epoch).

Hyperparamètres Sur l'ensemble de nos différents modèles, nous avons eu des résultats qui sont relativement proches pour les données de test malgré les différences sur les hyperparamètres. La précision tourne aux alentours des 80% et la perte aux alentours des 0,4. Il est alors à noter que les hyperparamètres influencent peu sur le résultat final du modèle ; que ce soit un LSTM de 64 variables ou un de 128, le résultat sera similaire. Cette conclusion n'est valable que dans le cas des modèles expérimentés dans ce travail.

Améliorations

Le premier point à noter serait probablement l'absence de classe neutre. Son importance a été négligée dans la recherche comme le montrent (Vryniotis) et (Koppel and Schler 2006). Elle est pourtant importante au niveau sémantique. En effet, une phrase peut être positive ou négative à un certain degré, c'est-à-dire que l'on peut échelonner la positivité. Par exemple en choisissant de rajouter des classes *très positif* ou *très négatif*. Ou encore en proposant une approche différente avec une régression au lieu d'un classificateur. Dans tous les cas, un corpus présentant des données neutres serait intéressant car une phrase neutre ne propose aucun élément de positivité ou négativité, il est donc inutile d'essayer de le positionner sur une telle échelle.
Un autre point qui aurait pu être étudié est le traitement des mots doublons comme "bye" et "byyyyyyye" qui est assez répandu dans le language commun et donc particulièrement avec twitter. Il pourrait être intéressant de voir si les compter comme des mots uniques n'apporte pas du bruit dans l'apprentissage de notre modèle.
En dernier point, compte tenu du fait que nous n'avions pas beaucoup de moyens de calcul et de temps, nous n'avons pas pu explorer autant de paramètres que nous aurions voulu. Il aurait pu être intéressant de traiter des modèles sans convolution ou bien en augmentant le nombre de paramètres dans les LSTMs.

Conclusion

Le but ultime de ce travail a été d'élaborer des modèles d'apprentissage machine pour évaluer la positivité d'un tweet. Nous avons utilisé le LSTM : ce qui a de mieux à l'heure actuelle dans l'état de l'art. Le travail a été effectué sur les données Sentiment140 qui avaient une taille raisonable pour que les expériences pussent se réaliser sur les machines de notre école. Les résultats obtenus nous ont ainsi permis de faire quelques conclusions que nous avons annoncé dans le paragraphe "Analyses". Enfin, les résultats que nous avons obtenus sont proches de ceux du travail effectué en 2009 sur les mêmes données (Go, Bhayani, and Huang 2009).

References

Alec Go, Richa Bhayani, L. H. 2009. Twitter sentiment classification using distant supervision.

Baziotis, C. ; Pelekis, N. ; and Doulkeridis, C. 2017. Datastories at semeval-2017 task 4 : Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 747–754. Vancouver, Canada : Association for Computational Linguistics.

Bo Pang, Lillian Lee, S. V. 2002. Thumbs up ? : sentiment classification using machine learning techniques.

Chung, J. ; Gülçehre, Ç. ; Cho, K. ; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555.

Cliche, M. 2017. Bb_twtr at semeval-2017 task 4 : Twitter sentiment analysis with cnns and lstms. *CoRR* abs/1704.06125.

Duyu Tang, Bing Qin, T. L. 2015. Document modeling with gated recurrent neural network for sentiment classification.

Go, A. ; Bhayani, R. ; and Huang, L. 2009. Twitter sentiment classification using distant supervision. 150.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Comput.* 9(8) :1735–1780.

Koppel, M., and Schler, J. 2006. The importance of neutral examples for learning sentiment. 22 :100–109.

Mikolov, T. ; Sutskever, I. ; Chen, K. ; Corrado, G. ; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. *CoRR* abs/1310.4546.

Pennington, J. ; Socher, R. ; and Manning, C. D. 2014. Glove : Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Rosenthal, S. ; Farra, N. ; and Nakov, P. 2017. SemEval-2017 task 4 : Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17. Vancouver, Canada : Association for Computational Linguistics.

Vryniotis, V. The importance of neutral class in sentiment analysis. `http://blog.datumbox.com/the-importance-of-neutral-class-in-sentiment-anal` Accessed : 2018-03-26.

Xiaowen Ding, Bing Liu, P. S. Y. 2008. A holistic lexicon-based approach to opinion mining.