Analyse d'opinions de tweets par réseaux de neurones convolutionnels

Jean-Marc Marty[1], Guillaume Wenzek[1], Eglantine Schmitt[1,2], Jocelyn Coulmance[1]
(1) Proxem, 105 rue La Fayette, 75010 PARIS
{jmm, guw, egs, joc}@proxem.com
(2) UTC, Compiègne
eglantine.schmitt@utc.fr

Résumé. La tâche d'analyse d'opinions consiste à détecter la polarité d'un texte (du plus négatif au plus positif). Nous présentons dans cet article un réseau de neurones permettant de trier de manière faiblement supervisée un ensemble de tweets en trois catégories : négatif, neutre ou positif. L'architecture du modèle est celle d'un réseau convolutionnel à trois couches mises en parallèles où chaque couche détecte des caractéristiques différentes. Le réseau est alimenté par des vecteurs-mots appris sur un ensemble de corpus dont la Wikipédia française, sans nécessiter d'informations linguistiques. En comparant cette approche avec un ensemble de techniques classiques alimentées par des sacs de mots, nous obtenons des résultats en moyenne 25% supérieurs en macro-précision.

Abstract.
Convolutional Neural Network for Twitter Sentiment Analysis
Sentiment Analysis is a common task in natural language processing that aims to detect polarity of a text document (from the most negative to the most positive). We introduce in this article a neural network that classifies in a weakly supervised fashion a set of tweets in three classes : negative, neutral or positive. The architecture of the model is that of a convolutional neural network with three parallel layers where each layer detects distinct features. The network is fed with word embeddings learned on a set of corpus among which the French Wikipedia with few linguistic informations. This model achieves a macro-precision in average 25% higher than classical methods based on bag of words.

Mots-clés : Analyse d'opinions, réseaux de neurones convolutionnels, Twitter.

Keywords: Sentiment Analysis, Convolutional Neural Network, Twitter.

1 Introduction

1.1 Analyse d'opinions pour Twitter

En quelques années, Twitter est devenu la plateforme de microblogage la plus étudiée dans la recherche. L'intérêt pour cet objet est dû à une combinaison de facteurs. D'une part, contrairement à Facebook, les contenus publiés par les utilisateurs y sont majoritairement publics et accessibles, bien que de façon restreinte, par une interface de programmation (API). Ainsi Facebook compte 1,441 milliards d'utilisateurs actifs mensuellements contre 302 millions sur la même période pour Twitter (Statista 2015a ; Statista 2015b) mais les travaux fondés sur l'analyse de grands volumes de données sont principalement dus aux équipes de Facebook elles-mêmes (Kramer *et al.*, 2014). D'autre part, Twitter bénéficie d'une forte notoriété (91% en France en mai 2014 d'après Ipsos) et d'une certaine mythologie selon laquelle il constituerait le pouls ou le miroir des sociétés. Malgré les difficultés posées, notamment en traitement automatique du langage, par la brièveté des messages qui y sont publiés, Twitter fait ainsi aujourd'hui l'objet de centaines de publications chaque année (Fausto & Aventurier, 2015) et serait ainsi devenu l'équivalent d'un organisme modèle en biologie, polarisant l'analyse de données issues de réseaux sociaux en ligne vers un exemple unique (Tufekci, 2014).

Outre des tâches plus typiques des sciences humaines et sociales telles que l'analyse de discours après échantillonnage, les données de Twitter mobilisent un large spectre de techniques de fouille de données et de texte appliqués à des volumes massifs d'information (big data). Parmi les tâches les plus fréquentes on peut citer l'analyse de réseaux sociaux, l'analyse de séries temporelles et l'analyse de sentiment ou d'opinions (Schmitt, 2015). Cette dernière est particulièrement exigeante

du fait de la brièveté des messages postés sur Twitter (les tweets) qui conduit elle-même à de nombreuses abréviations et à un jargon propre, distinct de la langue générale. L'ironie semble également particulièrement présente sur Twitter. Il est donc difficile de mettre en œuvre tels quels les outils conçus pour des corpus journalistiques ou scientifiques tels que SentiWordNet (Baccianella *et al.*, 2010) ou le LIWC (Pennebaker *et al.*, 2001). Pour Twitter comme pour d'autres données textuelles, la classification supervisée à partir d'un corpus annoté permet de placer la question de la modélisation du sentiment ou de l'opinion en amont de l'étape de classification ; le sentiment ou l'opinion n'est pas défini(e) formellement comme le ferait un psychologue mais empiriquement et intuitivement à partir d'exemples constitués par des humains.

1.2 Méthodes classiques

L'analyse d'opinion est un problème ouvert en traitement automatique du langage, qui présente des difficultés en termes de modélisation. Les méthodes classiques (arbres de décisions, réseaux bayésiens ou SVM chez (Pak & Paroubek, 2010; Baucom *et al.*, 2013; Dodds *et al.*, 2011; Psomakelis *et al.*, 2015)), utilisant des sacs de mots, ont des taux de précision au mieux de l'ordre de 70% sur une tâche de classification ternaire (positif/neutre/négatif).

L'approche classique consistant à repérer la connotation positive ou négative d'un terme et d'en déduire l'opinion d'une phrase ou d'un texte n'est pas suffisante ; il est également nécessaire a minima d'analyser l'ordre des mots et les relations qu'ils entretiennent. Ainsi la phrase «Le plat était bon mais pas très sucré» n'a pas le même sens que «Le plat était sucré mais pas très bon». Pourtant ces deux phrases auraient la même représentation dans un sac de mot.

Au delà de ce premier problème, les structures plus complexes qu'une construction sujet-verbe-complément sont très courantes et difficiles à capturer par des modèles semi-supervisés. Parmi elles la négation : «Ce film n'était pas terrible» et la double négation : «Ce film était pas mal». Ces phénomènes peuvent être capturés par des règles linguistiques. Mais ces règles sont longues à écrire, nécessitent une bonne connaissance de la langue et sont difficilement exhaustives.

Par ailleurs, Twitter possède sa syntaxe et ses règles linguistiques propres, qui doivent être prises en compte. Certains bons résultats récemment obtenus prennent généralement en entrée, en plus des sacs de mots, des *features* (caractéristiques) fabriquées à la main, comme celles introduites par (Mohammad *et al.*, 2013) qui ont été conçues spécialement pour Twitter.

Notre approche dans ce papier est d'effectuer l'analyse d'opinions de la façon la plus faiblement supervisée possible, en injectant le moins possible d'information linguistique, de sorte que notre modèle soit facilement adaptable à plusieurs langues et à d'autres types de syntaxe.

2 Introduction aux réseaux de neurones pour le TAL

2.1 Représentation vectorielle des mots

Avant de construire un modèle capable de capturer le sens d'une phrase, il paraît important de capturer dans un premier temps le sens d'un mot. Dans l'approche de type sac de mots, on considère que chaque mot ou stemme (quand on utilise une racinisation) a un sens unique. Deux mots sont alors soit identiques soit différents. Cependant, il paraît important que notre système soit capable de comprendre que certains mots ont un sens proche, comme «bleu» et «cyan», tandis que d'autres ont un sens éloigné, comme «bleu» et «chaise». L'idéal serait de pouvoir utiliser une distance mathématique sur les représentations de nos vecteurs-mots qui nous donne une idée de la distance sémantique entre les mots.

Considérer que le sens d'un terme se déduit de son contexte d'utilisation est une idée courante en sciences du langages. Par exemple, la signification d'une proposition chez le second Wittgenstein vient de sa valeur d'usage ; chez François Rastier, la sémantique d'un texte est construite de manière interprétative au moment de la lecture et ne peut être déduite comme un calcul à partir de sa syntaxe et de dictionnaire. En 1990, (Church & Hanks, 1990) proposait une métrique pour trouver des mots avec un sens proche : la "Pointwise Mutual Information" (PMI). Le but était de trouver les actions possible avec un téléphone. Pour cela, il calcula la PMI du mot "phone" avec chaque verbe de langue anglaise. La PMI des mots x et y se définit à partir du nombre d'occurrences de chacun ($\#(x)$ et $\#(y)$), du nombre de co-occurrences $\#(x, y)$ au sein d'une fenêtre de mots, et de la taille du corpus N suivant la formule :

$$\mathrm{PMI}(x, y) = \log \frac{N\#(x, y)}{\#(x)\#(y)}$$

sit by	11.78
disconnect	9.48
answer	8.80
hang up	7.87
tap	7.69
pick up	5.63
return	5.01
be by	4.93
spot	4.43
repeat	4.39

TABLE 1 – PMI entre le mot «phone» et les verbes qui lui sont le plus souvent associés.

bleu	1,00
rouge	0,91
jaune	0,89
violet	0,87
gris	0,87
blanc	0,85
mauve	0,85
couleur bleue	0,85
bleu ciel	0,84
marron	0,84

TABLE 2 – Les mots les plus proches du mot «bleu» avec leur score de similarité cosinus.

Cette formule permet de trouver les verbes anglais les plus sémantiquement proches du mot «phone» avec les scores indiqués en table 2.1.

Il est important de noter que nous nous ne voulons sûrement pas traiter les mots "phone" et "pick up" de façon similaire, mais en revanche nous voulons traiter "answer" de la même façon que "pick up". Nous avons donc une métrique qui nous permet de dire que deux mots sont «proches» dans un certain contexte (ici «phone» joue ce rôle de contexte). Maintenant que nous disposons d'une métrique formellement définie, nous pouvons construire des vecteurs qui nous permettent de capturer cette information. On voudrait être capable de reconstruire à partir de la représentation d'un mot et de la représentation d'un contexte la PMI de ce mot par rapport à ce contexte. Nous nous imposons les contraintes suivantes :

1. Chaque mot x se verra associer deux vecteurs : $\vec{x}$ pour x en tant que mot et $\overline{x}$ pour x en tant que contexte.

2. Pour un mot x et un contexte y on veut $\mathrm{PMI}(x, y) = \vec{x} \cdot \overline{y}$

Cela revient à factoriser la matrice de PMI de taille $V \times V$, où V est la taille du vocabulaire, en une matrice de vecteurs-mots de taille $d \times V$ et une matrice de vecteurs-contextes $V \times d$. Ici d est un paramètre que l'on choisit arbitrairement comme taille de nos vecteurs. Plus d est grand plus on peut reconstruire correctement la matrice de PMI. En revanche un d trop grand rend la manipulation des vecteurs peu aisée. En pratique on prend d entre 100 et 500.

Le problème de factorisation de matrice est connu de longue date, et cette technique de fabrication de mot a déjà été utilisée. Toutefois cette méthode est longue et coûteuse car la matrice est de grande taille. De plus les algorithmes classiques de factorisation traitent indifféremment chaque ligne de la matrice alors que nous voudrions favoriser les lignes correspondant à des mots fréquents.

(Mikolov *et al.*, 2013) introduit l'algorithme Word2Vec permettant de calculer des vecteurs-mots sur des gros corpus en des temps raisonnables. Mikolov a rendu public des vecteurs-mots à 300 dimensions pour 3 millions de mots et bigrammes anglais obtenus à partir d'un corpus de 100 milliards de mots extrait de Google News. (Levy & Goldberg, 2014) montre que cette méthode revient justement à faire une factorisation de la matrice de PMI qui favorise une bonne reconstruction pour les mots fréquents. Word2Vec a l'avantage d'être très simple à implémenter et d'offrir un temps de calcul raisonnable ; il faut en effet environ une heure pour entraîner des vecteurs-mots à partir de la Wikipédia française, sur un serveur disposant de 16 cœurs.

De plus, Word2Vec calcule de bonnes représentations même pour les termes peu fréquents ; ceci nous permet d'entraîner un vecteur pour chaque forme fléchie, sans nécessiter de racinisation.

Nous avons entraîné de tels vecteurs sur la Wikipédia française en utilisant cet algorithme. Les proximités des vecteurs obtenus - au sens de la distance euclidienne - reflètent bien leurs proximités sémantiques intuitives. Ainsi les dix termes les plus proches du mot «bleu» sont indiqués en table 2. Les scores reflètent le fait que «bleu», «rouge» et «jaune» apparaissent dans des contextes similaires en français. On voit que les vecteurs-mots ainsi appris permettront d'injecter dans notre modèle une certaine connaissance du monde.

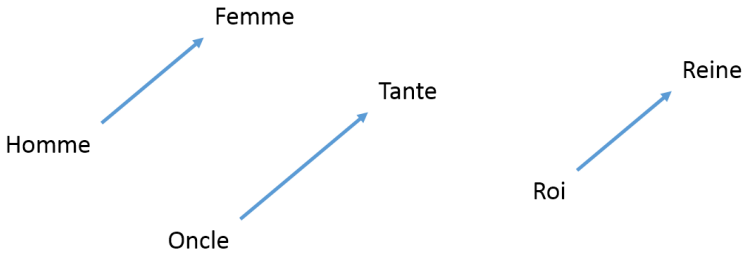

FIGURE 1 – Différences de vecteurs capturant la relation «passer au féminin». Figure extraite de (Mikolov *et al.*, 2013)

2.2 Composition des vecteurs-mots

Une autre propriété intéressante de ces vecteurs est l'additivité. On remarque que l'espace des vecteurs-mots s'est doté lors de l'apprentissage d'une structure additive. Ceci est particulièrement intéressant pour pouvoir transférer des relations. On constate ainsi que le vecteur le plus proche de $\vec{roi}+\vec{homme}-\vec{femme}$ est $\vec{reine}$. Ceci nous permet d'écrire la relation : $\vec{roi} - \vec{reine} \simeq \vec{homme} - \vec{femme}$. Ainsi le vecteur $\vec{homme} - \vec{femme}$ capture le changement de genre grammatical et sémantique. La figure 1 illustre ce phénomène.

Les vecteurs-mots capturent donc une information hiérarchisée. Ils sont localement regroupés par proximité sémantique, et ces groupes sont organisés selon des relations de type "passer au féminin". On a ainsi à la fois une information sémantique et syntaxique incluse dans les vecteurs-mots.

Une autre façon intéressante de composer les vecteurs-mots est la projection. Si on regarde par exemple le vecteur du mot «orange», on se rend compte qu'il est proche de $\vec{bleu}$ et des autres couleurs. En enlevant la composante du mot $\vec{orange}$ selon l'axe $\vec{bleu}$ on trouve un vecteur dont le plus proche voisin est $\vec{olive}$ et d'autres fruits et légumes. Ceci revient mathématiquement à projeter $\vec{orange}$ sous le sous-espace orthogonal à $\vec{bleu}$. L'opérateur de projection $\%$ est défini ainsi :

$$\vec{x} \,\%\, \vec{y} = \vec{x} - \frac{\vec{x} \cdot \vec{y}}{\vec{y} \cdot \vec{y}} \vec{y}$$

On peut donc écrire : $\vec{orange} \,\%\, \vec{bleu} \simeq \vec{pomme}$. Ceci montre que le vecteur $\vec{orange}$ porte l'ambiguïté du mot «orange», l'opérateur $\%$ permet de retirer un des sens de «orange», ici celui de couleur, et on trouve un autre sens du mot «orange» celui de fruit. Les vecteurs-mots arrivent donc à capturer beaucoup d'information même quand les mots sont ambigus.

Pour passer d'une représentation vectorielle du mot à une représentation vectorielle de la phrase, une idée naturelle est de combiner les vecteurs correspondants aux mots de la phrase. On peut se contenter d'additionner les vecteurs-mots contenus dans la phrase ou, mieux, effectuer une moyenne des vecteurs-mots de la phrase pondérée par les TF-IDF de chaque mot. Cependant ces deux méthodes ne capturent pas l'ordre des mots dans la phrase. On peut aussi s'appuyer sur des méthodes conservant une partie de la structure de la phrase. Par exemple, (Wu *et al.*, 2014) distingue d'une part les vecteurs des mots correspondant à des agents et ceux des mots correspondants à des patients, et somme les produits des couples (agent, patient). Une fois les vecteurs-mots combinés en un vecteur pour la phrase, celui-ci peut être fourni à un système de classification tel un SVM ou un réseau de neurones. Cette méthode combinée à un SVM permet à Wu de distinguer les discours de George W. Bush de ceux de Barack Obama avec une précision de 85%.

Rappelons que nous souhaitons injecter un minimum de connaissances linguistiques dans notre modèle. Nous souhaitons donc que notre système apprenne le plus automatiquement possible la bonne façon de combiner les vecteurs-mots pour la tâche d'analyse d'opinions. Nous avons vu que les vecteurs-mots ont des propriétés additives et projectives. Il parait donc logique de choisir un modèle qui puisse effectuer des opérations linéaires sur les vecteurs-mots, comme un réseau de neurones. Nous décrivons dans la section suivante pourquoi nous nous tournons vers un réseau de neurones convolutionnel.

3 Description du modèle utilisé

3.1 Introduction aux réseaux convolutionnels

Un réseau convolutionnel est un type particulier de réseau de neurones.

Chaque couche - dite de *convolution* - balaye l'ensemble de la couche précédente en appliquant à chaque petite région un même traitement local. Dans le cas d'un texte, les régions sont les n-grammes de la phrase. Pour une image, les régions sont classiquement des carrés de pixels ; la convolution permet alors par exemple d'y détecter des contours ou de flouter l'image. Ce filtre balaye ainsi l'ensemble de la sortie de la couche précédente et a pour rôle de détecter des propriétés locales. En superposant de telles couches de convolution les unes au-dessus des autres, on espère détecter des propriétés de plus en plus globales.

Cette méthode s'inspire de l'organisation des neurones du cortex visuel. Ces cellules sont sensibles à des petites régions du champ visuel et en recouvrent l'ensemble afin d'exploiter les propriétés locales de l'image reçue par les yeux. Ces réseaux, introduits par (LeCun & Bengio, 1995), semblent donc adaptés pour la vision par ordinateur et continuent à obtenir des résultats remarquables dans ce domaine (Krizhevsky *et al.*, 2012).

Entre chaque couche de convolution, on rajoute une couche - dite de *pooling* - qui ne conserve que les k plus grandes valeurs, où k est un hyperparamètre à sélectionner (c'est-à-dire un paramètre choisi arbitrairement). Cela a deux intérêts principaux : d'une part, cela diminue le nombre de calculs à effectuer. D'autre part, cela permet de calculer une forme d'invariant pour la translation, ce qui est effectivement ce qu'on recherche lors de la détection d'images par exemple.

L'intérêt de cette méthode dans le cas du TAL est principalement de détecter des motifs récurrents dans une phrase. On pourrait par exemple détecter l'intensification : «très bien», «vraiment mauvais», etc. On pourrait également détecter la négation : «j'aime pas», «pas mal», etc. Grâce à la couche de pooling, on peut également détecter des n-grams à distance : «autant ..., autant ...» ou encore «j'aime bien ..., mais». (Blunsom *et al.*, 2014) cumule des couches de convolution et de pooling pour faire de l'analyse sémantique et (Kalchbrenner *et al.*, 2014) utilise la même approche pour la modélisation de phrases.

3.2 Architecture du réseau utilisé

Dans le travail suivant, nous nous inspirons du modèle proposé par (Kim, 2014). Il utilise un réseau peu profond comprenant 3 couches de convolution, non pas successives mais mises en parallèle, avec des filtres de taille 3, 4 et 5 sur la longueur de la phrase et de taille 300 sur la taille du vecteur-mot (ce qui correspond en fait à la longueur des vecteurs). Une couche de pooling est placée après chaque couche de convolution. [1]

Enfin, pour que l'algorithme prenne une décision finale sur le sentiment de la phrase, on lui ajoute une couche de neurones avec la fonction d'activation softmax (notée σ), définie comme tel :

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{k=1}^{|C|} e^{z_k}}$$

où $|C|$ est le nombre de classes. Le softmax permet de transformer la sortie du réseau en une distribution de probabilité sur les différentes catégories possible, en les normalisant.

Avant la couche de softmax, on rajoute également une couche de dropout, technique popularisée par (Srivastava *et al.*, 2014), qui permet d'améliorer la généralisation du modèle, mais rend l'apprentissage un peu plus long. Le réseau prend en entrée la matrice représentant la phrase avec pour chaque colonne la représentation vectorielle du mot correspondant (voir figure 2).

Kim a effectué un certain nombre d'expérimentations avec cette architecture :

— Il tente d'abord d'apprendre les vecteurs-mots en partant de zéro ;
— Il utilise ensuite directement les vecteurs appris par Mikolov en les laissant inchangés ;
— Enfin, il utilise les vecteurs de Mikolov et les ré-apprend au cours de l'apprentissage.

1. On peut se poser la question de l'utilité du calcul de filtres de taille 3 et 4 vu qu'on pourrait théoriquement détecter les mêmes caractéristiques avec un filtre de taille 5 et une couche de pooling. Cependant, on constate empiriquement que le temps d'apprentissage est alors plus court. Le réseau de neurones «sait» déjà quoi détecter, ce qui accélère les choses.

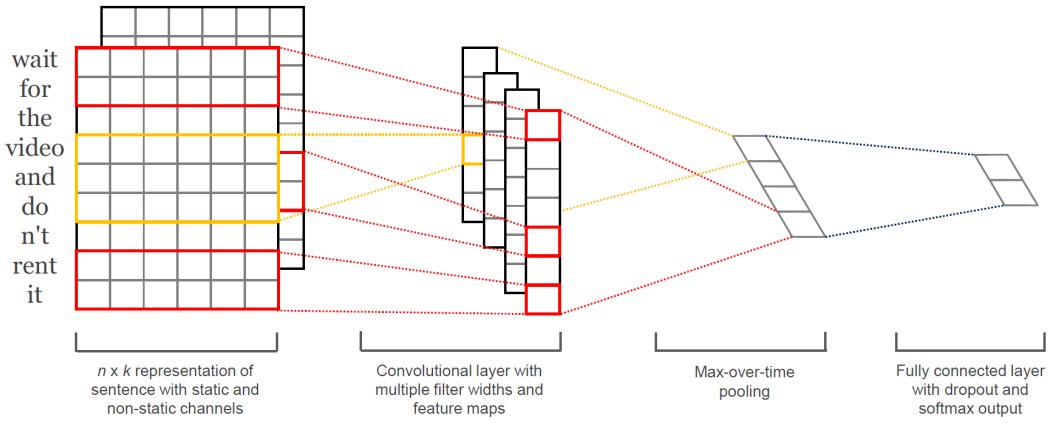

FIGURE 2 – Le réseau utilisé par (Kim, 2014)

C'est avec cette dernière approche qu'il obtient ses meilleurs résultats. En effet, les vecteurs de Mikolov sont de bonne qualité mais ne sont pas appris dans le but particulier de l'analyse d'opinions et possèdent donc naturellement du bruit. En s'inspirant de ce résultat, nous voulons conserver la possibilité pour le réseau de re-modifier les vecteurs de Mikolov. En revanche, nous ne voulons pas modifier les vecteurs précédemment appris car nous voulons pouvoir les réutiliser pour une autre tâche. De plus, nous souhaiterions diminuer la taille des vecteurs pour minimiser le temps d'apprentissage. La solution que nous avons mise en œvre est de rajouter une couche de neurones en entrée qui va faire passer la taille des vecteurs de 300 à 30, tout en allant récupérer l'information pertinente pour l'analyse d'opinions.

4 Jeu de données et protocole expérimental

4.1 Présentation et pré-processing du jeu de données

Le jeu de données du Défi Fouille de Texte DEFT 2015 est un ensemble de 15 000 tweets en français portant sur le thème de l'écologie. Ces tweets sont triés en trois classes (positif, neutre, négatif).

Toujours dans l'optique d'injecter le moins d'information linguistique *à la main*, nous n'effectuons que très peu de pré-traitements. Nous retirons les URLs ainsi que les mentions Twitter (de type `@proxem`). Nous appliquons enfin un tokenizer standard pour le français. Nous laissons le modèle apprendre les caractéristiques utiles à l'analyse d'opinions avec une approche non-supervisée.

4.2 Hyperparamètres du modèle

Nous utilisons les *rectified linear units* introduites dans (Nair & Hinton, 2010) comme fonctions d'activation des neurones. Une fois passée la première couche faisant passer la taille des vecteurs-mots de 300 à 30, on effectue l'opération de convolution avec 100 filtres linéaires de taille 3, 100 filtres linéaires de taille 4, 100 filtres linéaires de taille 5.

Le résultat du passage de chacun de ces filtres est ce qu'on appelle un *feature map* qui sera un vecteur de taille égale à celle de la phrase en entrée. Comme cette taille varie selon les tweets et que nous avons besoin d'une taille fixe pour appliquer la couche de softmax, l'opération de pooling va sélectionner la composante maximale de ce vecteur. Grâce à cette opération, nous forçons le réseau à sélectionner la caractéristique la plus importante de la phrase et nous obtenons un vecteur de taille fixe (taille que nous avons choisie égale à 1) pour chaque filtre. Une fois passée la couche de pooling notre réseau n'a donc plus connaissance des positions relatives des différentes expressions capturées par chaque filtre.

A l'issue de ces couches de convolution, nous obtenons finalement un vecteur concaténé de taille 300 (correspondant au nombre total de filtres choisi), que nous donnons en entrée d'une couche de softmax, à laquelle on rajoute une couche de dropout avec un coefficient de dropout de 0.5. Nous rajoutons également une contrainte sur la norme L^2 des poids du

modèle [2] pour qu'elle ne dépasse pas 3 lors de la descente de gradient.

L'apprentissage se fait par descente de gradient stochastique, avec des paquets de taille 50 et la règle de descente Adadelta (Zeiler, 2012). Enfin, nous arrêtons l'apprentissage en suivant l'évolution des performances par cross-validation.

5 Résultats

5.1 Comparaison avec un SVM utilisant des sacs de mots

Nous avons comparé les résultats obtenus dans notre expérience avec un certain nombre de modèles utilisant des sacs de mots et la pondération TF-IDF. Certains de ces modèles sont linéaires (régression logistique, SVM à noyau linéaire), d'autres non (forêts aléatoires). La métrique utilisée est la macro-précision, définie comme tel :

$$P_{macro} = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{VP_i}{VP_i + FP_i}$$

où $|C|$ est le nombre de classes, VP_i est le nombre de vrais positifs pour la classe i et FP_i est le nombre de faux positifs pour la classe i.

Modèles	Macro-Précision
Réseau bayésien	44.7
SVM	41.8
Forêt aléatoire	46.0
Régression logistique	41.4
Réseau Convolutionnel	**69.9**

TABLE 3 – Résultats obtenus

Bien que peu profond et alimenté de peu d'informations linguistiques, notre réseau de neurones surpasse les techniques classiques sur cette tâche comme le montre la table 3.

5.2 Durée d'apprentissage

Un autre point positif est que grâce à notre première couche qui réduit la taille de nos vecteurs-mots et du fait que le réseau soit peu profond, notre algorithme converge assez rapidement, puisqu'en 30 minutes de calcul sur CPU (serveur avec 16 cœrs) nous arrivons presque au score obtenu plus haut.

6 Conclusion

Nous avons présenté dans cet article un modèle d'analyse d'opinions faiblement supervisé, se fondant sur un simple réseau convolutionnel de neurones à une couche de convolution et prenant en entrée des vecteurs-mots appris sur la Wikipedia française ainsi que d'autres sources selon la méthode introduite par Mikolov. Nous avons obtenu des résultats encourageants sur un jeu de données de tweets en français sur le thème de l'écologie. Notre approche pourrait être évaluée sur d'autres thèmes ou en monde ouvert. Ces résultats nous conduisent à poursuivre notre travail sur les vecteurs-mots et les réseaux convolutionnels. Nous sommes par ailleurs confiants quant à l'apport des réseaux de neurones récurrents sur lesquels nous travaillons également.

2. $\forall \vec{x} \in R^n,\ L^2(\vec{x}) = \sum_{0 \leq i < n} x_i^2$

Références

BACCIANELLA S., ESULI A. & SEBASTIANI F. (2010). Sentiwordnet 3.0 : An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, p. 2200–2204.

BAUCOM E., SANJARI A., LIU X. & CHEN M. (2013). Mirroring the real world in social media : twitter, geolocation, and sentiment analysis.

BLUNSOM P., DE FREITAS N., GREFENSTETTE E., HERMANN K. M. *et al.* (2014). A deep architecture for semantic parsing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing* : Proceedings of the ACL 2014 Workshop on Semantic Parsing.

CHURCH K. W. & HANKS P. (1990). Word association norms, mutual information, and lexicography. *Computational linguistics*, **16**(1), 22–29.

G. DIAS, Ed. (2015). *Actes de TALN 2015 (Traitement automatique des langues naturelles)*, Caen. ATALA, HULTECH.

DODDS P. S., HARRIS K. D., KLOUMANN I. M., BLISS C. A. & DANFORTH C. M. (2011). Temporal patterns of happiness and information in a global social network : Hedonometrics and twitter. *PloS one*, **6**(12), e26752.

FAUSTO S. & AVENTURIER P. (2015). Scientific literature on twitter as subject research : preliminary findings based on bibliometric analysis. In *Twitter for Research 2015*, p.1̃p.

KALCHBRENNER N., GREFENSTETTE E. & BLUNSOM P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv :1404.2188*.

KIM Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv :1408.5882*.

KRAMER A. D., GUILLORY J. E. & HANCOCK J. T. (2014). Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, **111**(24), 8788–8790.

KRIZHEVSKY A., SUTSKEVER I. & HINTON G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, p. 1097–1105.

LECUN Y. & BENGIO Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, **3361**, 310.

LEVY O. & GOLDBERG Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, p. 2177–2185.

MIKOLOV T., CHEN K., CORRADO G. & DEAN J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*.

MOHAMMAD S. M., KIRITCHENKO S. & ZHU X. (2013). Nrc-canada : Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (SEMSTAR'13)*.

NAIR V. & HINTON G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, p. 807–814.

PAK A. & PAROUBEK P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, volume 10, p. 1320–1326.

PENNEBAKER J. W., FRANCIS M. E. & BOOTH R. J. (2001). Linguistic inquiry and word count : Liwc 2001. *Mahway : Lawrence Erlbaum Associates*, **71**, 2001.

PSOMAKELIS E., TSERPES K., ANAGNOSTOPOULOS D. & VARVARIGOU T. (2015). Comparing methods for twitter sentiment analysis. *arXiv preprint arXiv :1505.02973*.

SCHMITT E. (2015). Elements for an epistemology of instrumentation and collaboration in Twitter data research. 1st International Conference on Twitter for Research.

SRIVASTAVA N., HINTON G., KRIZHEVSKY A., SUTSKEVER I. & SALAKHUTDINOV R. (2014). Dropout : A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, **15**(1), 1929–1958.

TUFEKCI Z. (2014). Big questions for social media big data : Representativeness, validity and other methodological pitfalls. *arXiv preprint arXiv :1403.7400*.

WU C., SKOWRON M. & PETTA P. (2014). Reading between the lines.

ZEILER M. D. (2012). Adadelta : an adaptive learning rate method. *arXiv preprint arXiv :1212.5701*.