LSE au DEFT 2018 : Classification de tweets basée sur les réseaux de neurones profonds

Antoine Sainson[1] Hugo Linsenmaier[1] Alexandre Majed[1] Xavier Cadet[1]
Abdessalam Bouchekif[2]
Laboratoire Système & Sécurité de l'EPITA (LSE), Paris, France
(1) prénom.nom@lse.epita.fr
(2) prénom.nom@epita.fr

RÉSUMÉ

Dans ce papier, nous décrivons les systèmes développés au LSE pour le DEFT 2018 sur les tâches 1 et 2 qui consistent à classifier des tweets. La première tâche consiste à déterminer si un message concerne les transports ou non. La deuxième, consiste à classifier les tweets selon leur polarité globale. Pour les deux tâches nous avons développé des systèmes basés sur des réseaux de neurones convolutifs (CNN) et récurrents (LSTM, BLSTM et GRU). Chaque mot d'un tweet donné est représenté par un vecteur dense appris à partir des données relativement proches de celles de la compétition. Le score final officiel est de 0.891 pour la tâche 1 et de 0.781 pour la tâche 2.

ABSTRACT

LSE at DEFT 2018 : Sentiment analysis model based on deep learning

In this article we present the contribution of the LSE at DEFT 2018 for task 1 and 2 which consists in classifying tweets. The goal of the first task was to identify if a tweet is related to transportation or not. The purpose of the second task was to identify the feelings associated to tweets. For both tasks we proposed models based on Convolutional (CNN) and Recurrent Neural Networks (LSTM, BLSTM and GRU). Each word in a given tweet is represented by a dense vector learned from data that is relatively similar to the one of the competition. The official score obtained according to the F-mesure is 0.891 for task 1 and 0.781 for task 2.

MOTS-CLÉS : Analyse d'opinions, plongement de mots, réseaux de neurones profonds, classification thématique.

KEYWORDS: Sentiment analysis, word embeddings, deep learning, text classification.

1 Introduction

La classification de textes est une des tâches importantes du traitement automatique du langage naturel (TALN). Elle consiste à attribuer une catégorie à un texte. La classification thématique et l'analyse des sentiments sont les applications les plus répandues auprès des entreprises. Par exemple, les fournisseurs d'actualités comme *Google Actualités* et *Yahoo Actualités* collectent des informations en provenance des sites d'information afin de les regrouper en thèmes. D'autres compagnies utilisent les tweets pour connaître l'opinion des utilisateurs sur un produit ou un service.

La compétition *DEFT 2018* (Paroubek *et al.*, 2018) propose deux tâches de classification de tweets français. La première consiste à classifier ces tweets selon leur thème (*transport* ou *inconnu*) et la deuxième à indiquer la polarité générale des tweets : *positif*, *négatif*, *neutre* ou *mixposneg*.

Avec les récents progrès en apprentissage profond, la performance des systèmes d'analyse de sentiments s'est considérablement améliorée. Par exemple, les auteurs de (Baziotis *et al.*, 2017) utilisent un réseau de neurones de type *BLSTM* (Bidirectional Long Short-Term Memory) avec des mécanismes d'attention tandis que (Deriu *et al.*, 2016) utilise des réseaux neuronaux convolutionnels (CNN). Les deux systèmes ont respectivement obtenu les meilleures performances lors des compétitions *SemEval* 2016 et 2017.

Les réseaux de neurones récurrents sont particulièrement adaptés aux données séquentielles de tailles variables. C'est pour ces raisons que les réseaux de neurones de type LSTM (Hochreiter & Schmidhuber, 1997), *BLSTM* (Schuster & Paliwal, 1997) et *GRU* (Cho *et al.*, 2014) sont les plus utilisés dans le traitement automatique du langage naturel (*e.g.* traduction automatique, analyse des sentiments, chatbot). Les CNNs ont l'avantage de prendre en compte le contexte des données d'entrée, ce qui permet d'avoir de bonnes performances non seulement dans l'analyse d'images, mais aussi dans l'analyse des données textuelle.

Cet article présente notre modèle d'analyse des sentiments basé sur la combinaison de quatre réseaux de neurones profonds. Ces derniers, prennent en entrée la représentation vectorielle des mots (*word embeddings*). Aucun corpus annoté ou lexique externe n'ont été utilisés, les données proviennent uniquement des corpus de test fournis.

L'article est structuré comme suit : dans la section 2, on décrit les prétraitements effectués, la représentation vectorielle des mots ainsi les différents modèles. La section 3 présente les résultats obtenus dans les deux tâches. Enfin, des expériences complémentaires sont décrites dans la section 4.

2 Systèmes Proposés

Dans cette section, nous présentons les trois étapes de notre approche :

1. **Prétraitements :** pour filtrer les imperfections des tweets.
2. **Word embeddings :** pour donner une représentation vectorielle des mots.
3. **Apprentissage des modèles :** pour classifier un tweet selon le thème ou le sentiment exprimé.

2.1 Prétraitements

Pour l'analyse de sentiments, les tweets contiennent plusieurs sources de bruit. L'étape de prétraitement consiste à les préparer pour un traitement automatique efficace.

2.1.1 Lemmatisation

Les mots d'une langue donnée sont accordés en genre, en nombre et en mode (indicatif, impératif...). Le rôle d'un lemmatiseur est de ramener le mot à sa forme canonique (*i.e.* les verbes à l'infinitif et les autres mots au masculin singulier). Ce processus permet de réduire la taille du vocabulaire et d'uniformiser nos tweets. Dans ce travail, nous avons utilisé l'outil *TreeTagger* [1].

1. http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

2.1.2 Normalisation

Cette étape consiste à convertir tous les mots en minuscule, supprimer les informations parasites telles que les URLs, les emails, les dates, les pseudos et les mots non porteurs de sens (*e.g.* le, de, ce, *etc.*). Néanmoins, les mots de négation comme *ne* et *pas* ont été retenus, car la négation dans une phrase peut inverser le sentiment du tweet (Das & Chen, 2001). Les smileys donnent aussi des informations supplémentaires sur les émotions de l'internaute. De ce fait, nous regroupons les smileys ayant une signification commune (tristesse, joie, colère, *etc.*).

2.2 Représentation vectorielle des mots

Les mots de chaque tweet sont représentés par des vecteurs de 104 et 102 dimensions respectivement pour les tâches 1 et 2. Ils ont été obtenus à partir caractéristiques suivantes :

2.2.1 Word embeddings

Le *word embedding* est une représentation de mots dans un espace à n dimensions apprise à partir de réseaux de neurones. Chaque mot est représenté par un vecteur de nombres réels capturant la sémantique des mots. Deux mots sont considérés comme similaires si leurs représentations vectorielles sont proches dans le même espace continu. Notre choix fut de construire notre propre représentation vectorielle, à partir de tweets récoltés sur internet, pour disposer d'une représentation vectorielle plus robuste et adaptée au problème que certains modèles trouvables sur le net. Nos sources proviennent d'1 million de tweets de différentes catégories relatives aux données de la compétition : $\#RATP(169k)$, $\#SNCF(453k)$, $\#IleDeFrance(90k)$, ainsi qu'une collection de tweets traitant de la politique française [2].

Les représentations vectorielles ont été obtenues en utilisant l'outil *Fasttext* [3], qui considère les mots comme des assemblages de caractères (Bojanowski *et al.*, 2016). Ainsi, les fautes d'orthographe n'influent que peu la différence de représentation des mots dans l'espace vectoriel engendré par le Word2vec. Nous fixons la taille maximale de la fenêtre du contexte à 4 et le nombre d'occurrences minimales à 5. Nous choisissons l'architecture *skip-gram* (Mikolov *et al.*, 2013) qui entraîne un réseau de neurones à prédire le contexte d'un mot donné. Les words embeddings utilisés dans ce travail sont de 100 dimensions.

2.2.2 Valence émotionnelle des mots

Pour la tâche 2, à partir des données d'apprentissage, nous associons à chaque mot un vecteur de valeurs comprises entre 0 et 1 représentant sa probabilité d'apparition dans les catégories suivantes : *positif*, *négatif*, *neutre* ou *mixposneg*.

2.2.3 Probabilité thématique des mots

Pour la tâche 1, à partir des données d'apprentissage, nous associons à chaque mot la probabilité d'apparition dans les deux catégories suivantes : *transport* et *inconnu*.

2. http://ideo2017.ensea.fr/projet-polititweets/
3. https://fasttext.cc/

2.3 Apprentissage des Modèles

Pour construire nos modèles, nous avons utilisé deux familles de réseaux de neurones profonds : convolutifs (CNN) et récurrents ($LSTM$, $BLSTM$ et GRU).

2.3.1 Réseaux de neurones récurrents

Les réseaux de neurones classiques supposent que toutes les entrées sont indépendantes les unes des autres, *c'est-à-dire* qu'ils ne prennent pas en compte l'ordre des entrées (dans notre cas, les mots). En revanche, les réseaux de neurones récurrents (RNN) traitent les données au fur et à mesure, tout en respectant l'ordre des entrées. Les RNNs sont principalement conçus pour modéliser des séquences et sont capables de mémoriser des informations passées.

Réseaux de neurones récurrents simples
Un RNN simple à l'instant t observe l'entrée $x^{(t)}$ (un mot dans un tweet) et met à jour la sortie de la couche cachée $h^{(t)}$ en prenant en compte le vecteur $h^{(t-1)}$ calculé à l'étape précédente. Du fait que chaque neurone conserve l'information des étapes précédentes, il est qualifié de *cellule de mémoire*. La sortie de la couche à l'étape temporelle t est donnée dans l'équation 1

$$h^{(t)} = f(W_{xh}\, x^{(t)} + W_{hh}\, h^{(t-1)} + b_h) \tag{1}$$

- f est la fonction d'activation.
- $x^{(t)} \in \mathbb{R}^{\mathrm{d}}$ est le vecteur du mot à l'instant t
- $W_{xh} \in \mathbb{R}^{\mathrm{d,n_h}}$ est la matrice des poids entre l'entrée $x^{(t)}$ et la couche cachée h, sachant que n_h est le nombre de neurones dans la couche cachée.
- $W_{hy} \in \mathbb{R}^{\mathrm{n_h,n_o}}$ est la matrice entre la couche cachée et la sortie, sachant que n_o est le nombre de neurones dans la couche de sortie.
- $W_{hh} \in \mathbb{R}^{\mathrm{n_h,n_h}}$ est la matrice des poids entre la sortie des couches cachées courante et précédente.

Les RNNs simples souffrent de la disparition/explosion des gradients. En effet, plus le modèle est profond, plus les valeurs des gradients propagées vers les couches basses sont affaiblies (*i.e.* les poids deviennent de plus en plus petits). Par conséquent, la mise à jour par descente de gradient a un impact très limité et la convergence vers l'optimum global n'est pas garantie. L'opposé de la disparition des gradients peut se produire, c'est-à-dire que la mise à jour des poids entraîne des modifications trop importantes, ce qui fait diverger le réseau. L'utilisation des fonctions d'activation non saturantes comme *RELU* (rectified linear unit) ou sa variante *ELU* (Exponetial Linear Unit) pourront alléger ce phénomène.
Le RNN simple souffre d'un autre problème, celui de la disparation des premières entrées en mémoire, qui vient principalement de la taille de la séquence. Prenons le tweet "*j'ai adoré les trains de la ligne L, il manque un peu de nettoyage de temps en temps*". Si le RNN perd de l'information sur les premiers mots, le système pourra considérer le message comme ayant une polarité négative.

LSTM, GRU

Les réseaux de type LSTM et GRU sont des variations de RNN simple, capables d'apprendre les dépendances à long terme. Cette capacité vient des cellules de mémoire. Chaque cellule LSTM est liée, non seulement à $h^{(t-1)}$ mais également à un état de la cellule c de l'étape précédente qui joue le rôle de mémoire. $h^{(t)}$ peut être vu comme l'état à court terme et $c^{(t)}$ comme l'état à long terme (Géron, 2017). Une cellule LSTM contient :

- 2 entrées : l'ancien état de la cellule $c^{(t-1)}$ et le vecteur $h^{(t-1)}$
- 4 couches intégralement connectées.
- 3 types de porte : porte d'oubli (f), porte d'entrée (i) et porte de sortie (o). Ces portes utilisent la fonction d'activation *sigmoïde*.
- 2 sorties : $c^{(t)}$ et $h^{(t)}$.

Une cellule LSTM a la capacité d'éliminer, d'ajouter et de stocker des informations dans l'état à long terme $c^{(t)}$. Les équations ci-dessous illustre le calcul de $c^{(t)}$ et $h^{(t)}$

$$
\begin{aligned}
i^{(t)} &= \sigma(W_{xi}\, x^{(t)} + W_{hi}\, h^{(t-1)} + b_i) \\
f^{(t)} &= \sigma(W_{xf}\, x^{(t)} + W_{hf}\, h^{(t-1)} + b_f) \\
o^{(t)} &= \sigma(W_{xo}\, x^{(t)} + W_{ho}\, h^{(t-1)} + b_o) \\
g^{(t)} &= \tanh(W_{xg}\, x^{(t)} + W_{hg}\, h^{(t-1)} + b_g) \\
c^{(t)} &= f^{(t)} \otimes c^{(t-1)} + i^{t} \otimes g^{(t)} \\
h^{t} &= \tanh(c^{(t)}) \otimes o^{(t)}
\end{aligned}
$$

W_{xi}, W_{xf}, W_{xo}, W_{hi}, W_{hf}, W_{ho} et W_{hg} sont des matrices de poids.

b_i, b_f, b_o et b_g sont les biais de chacune des quatre couches.

$\otimes$ est le produit matriciel de Hadamard.

La porte d'oubli permet de filtrer les informations provenant du précédent état à long terme $c^{(t-1)}$. Puisqu'elle utilise la fonction d'activation sigmoïde qui sort des valeurs entre 0 et 1, et en les multipliant par $c^{(t-1)}$, la cellule LSTM filtre les informations de $c^{(t-1)}$. C'est la porte d'entrée $i^{(t)}$ et la sortie de la couche $g^{(t)}$ qui sont responsables de la mise à jour de l'état à long terme. La porte de sortie $o^{(t)}$ est chargée de sélectionner les informations de l'état à long terme à produire à la sortie de la cellule (dans h_t).

La cellule GRU est une version simplifiée de la cellule LSTM (*i.e.* contient moins de paramètres). À la différence des réseaux de type LSTM, dans les réseaux de type GRU :

- On trouvera uniquement deux types de portes : reset gate (r) et update gate (z).
- Les deux vecteurs d'états sont fusionnés en un seul vecteur h_t

L'équation ci-dessous illustre le calcul $h^{(t)}$

$$
\begin{aligned}
z^{(t)} &= \sigma(W_{xz}\, x^{(t)} + W_{hz}\, h^{(t-1)} + b_z) \\
r^{(t)} &= \sigma(W_{xr}\, x^{(t)} + W_{hr}\, h^{(t-1)} + b_r) \\
g^{(t)} &= \sigma(W_{xg}\, x^{(t)} + W_{hg}\, h^{(t-1)} + b_g) \\
h^{(t)} &= z^{(t)} \otimes h^{(t-1)} + (1 - z^{(t)} \otimes g^{(t)})
\end{aligned}
$$

BLSTM
Les réseaux de type BLSTM consistent à exécuter deux LSTM en parallèle : le premier réseau lit la séquence d'entrée de droite à gauche et le second réseau en sens inverse de gauche à droite. Chaque LSTM engendre une représentation cachée : $\vec{h}$ (un vecteur allant de gauche à droite) et $\overleftarrow{h}$ (un vecteur allant de droite à gauche) qui sont ensuite combinés afin de calculer la séquence de sortie. Dans notre problème, saisir le contexte des mots de chaque direction permet de mieux comprendre la sémantique d'un tweet.

2.3.2 Réseau neuronal convolutif

Une architecture de CNN typique consiste en la superposition de plusieurs couches de convolutions, mise en commun (*pooling*) et intégralement connectées. Dans la couche de convolution, chaque neurone est connecté à une région de l'entrée. L'opération de convolution consiste à appliquer une petite fenêtre de poids (également appelé noyau de convolution, filtre ou détecteur de caractéristiques), appliqués à des caractéristiques locales.
Nous avons implémenté avec Keras [4] des CNNs ayant respectivement 1, 2 et 3 couches de convolution pour lesquels nous avons fait varier divers paramètres.
Une autre variante du CNN consiste à appliquer une seule couche de convolution en entrée, en utilisant des noyaux de convolution de différentes tailles dans des réseaux séparés. Les vecteurs de caractéristiques de chaque réseau sont ensuite rassemblés pour n'obtenir qu'un vecteur de caractéristiques. Ce vecteur est ensuite donné en entrée à des couches connectées pour classer le tweet. Ce modèle figurait comme une option intéressante pour la classification de phrases (Zhang & Wallace, 2015).

3 Résultats et analyse

Nous avons opté pour l'utilisation de différents types de réseaux profonds : notre première approche fut de maximiser individuellement le score de chacun de ces réseaux et de mieux comprendre les données de test à partir de leurs sorties.

CNN : Il s'agit d'un CNN d'une couche de convolution de 64 neurones et une fenêtre de taille 5. On applique ensuite un max pooling global avant d'insérer une couche de 128 neurones, un *dropout* de 0.1 et la couche de sortie.

4. https://keras.io/

CNN 2 : Le modèle est composé de deux couches de convolutions de 64 filtres et d'une taille de fenêtre de 5. Un *max pooling* est appliqué après chaque convolution. On insert un *dropout* de 0.3 après la première couche de convolution. Enfin, on ajoute une couche dense de 64 neurones suivie par une couche de sortie de taille correspondante au nombre de classes.

CNN 3 : On définit 3 sous modèles avec les paramètres suivants : une couche de convolution de 64 neurones et des fenêtres de tailles 3, 4 puis 5. On y ajoute un *max pooling* global. Ces 3 modèles sont fusionnés grâce à la couche *Merge* de Keras. On ajoute par la suite un *dropout* de 0.5 suivi d'une couche dense de 64 neurones et de la couche de sortie.

BLSTM : La taille du LSTM est de 64 neurones (128 pour le BLSTM). On y ajoute un *dropout* de 0.5 suivi de la couche de sortie.

GRU : Une couche GRU de 64 neurones suivie de la couche de sortie.

LSTM : Une couche LSTM de 64 neurones suivie de la couche de sortie.

Combinaison 1 : Une combinaison de nos meilleurs modèles : les sorties de chaque modèle nourrissent un réseau de neurones de 3 couches cachées de 20, 20 et 10 neurones. Entre les 2 premières couches, on applique un *dropout* de 0.2.

Combinaison 2 : Une moyenne des prédictions de chaque modèle.

3.1 Analyse

3.1.1 Tâche 1

Le tableau 1 présente la répartition des données en fonction des deux catégories étudiées : *transport* et *inconnu*. Le corpus d'apprentissage contient 59990 tweets : 51.6% *transport* et 48.4% *inconnue*. Le corpus de test est composé de 7816 tweets : 50.4% *transport* et 49.6% *inconnue*

	Transport	Inconnu	Total
Dev.	30951	29039	59990
Test	3941	3875	7816

TABLE 1 – Proportion des données de test et d'entraînement de la tâche 1.

Type de réseau	Précision	F1-mesure
CNN	0.799	0.888
CNN 2	0.798	0.888
CNN 3	0.803	0.890
BLSTM	0.810	0.892
GRU	0.800	0.889
LSTM	0.795	0.886
Combinaison 1	0.767	0.868
Combinaison 2	**0.808**	**0.894**

TABLE 2 – Précision et F1-mesures des données de test de la tâche 1

Le tableau 2 présente les résultats obtenus à la tâche 1. En utilisant, les réseaux de neurones convolutifs

on obtient des performances similaires. Avec les réseaux de neurones récurrents la *F-mesure* prend des scores allant de 0.886 à 0.892. Le meilleur système est celui qui combine les 5 modèles : *CNN, CNN2, BLSTM, GRU et LSTMI* avec une F-mesure de 0.984%. Présenter les sorties des 5 modèles à un réseau de neurone dense (combinaison2) dégrade les performances de notre système de classification

La figure 1 met en avant le fait que notre modèle classifie correctement le plus souvent les tweets concernant les transports.

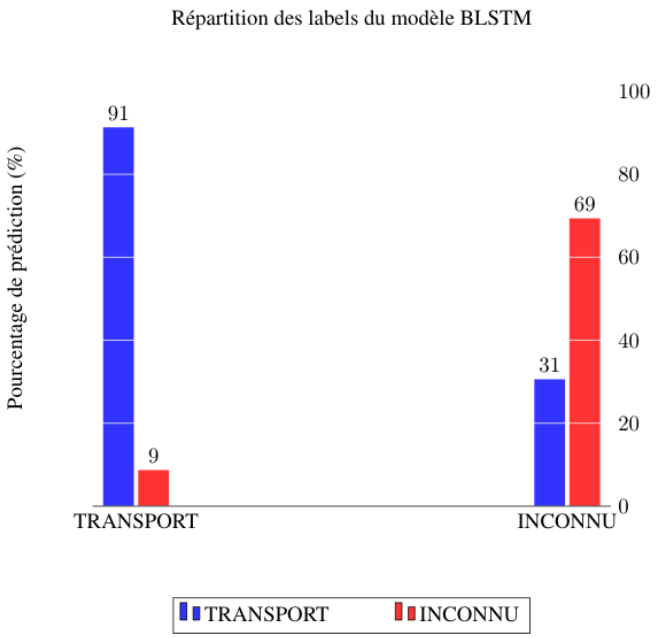

FIGURE 1 – Répartition des prédictions par label de référence du modèle BLSTM de la tâche 1

En effet, notre système détecte facilement les mots comme "bus" ou "métro" comme étant spécifiques aux tweets concernant les transports. Cependant, il arrive parfois que ces mots soient contenus dans des tweets labéllisés *non transport*, ce qui fausse complètement l'analyse du reste du tweet, tant ces mots ont une forte appartenance à la classe *transport*. Les mots les plus susceptibles d'induire en erreur notre système ont été regroupés dans le tableau 3 :

Mots-clés	occurrences
pas	327
métro	158
train	113
sncf	92
rer	88

TABLE 3 – Mots les plus apparents pour les erreurs d'annotations *non transport*

3.1.2 Tâche 2

Le tableau 4 présente la répartition des données en fonction des quatre catégories étudiées *positif*, *négatif*, *neutre* et *mixposneg*. Le corpus d'apprentissage est composé de 35455 tweets (20.7% *positif*, 37.0% *négatif*, 35.6% *neutre* et 6.7% *mixposneg*). Le corpus de test contient 3941 tweets (21.7%

	Positif	Négatif	Neutre	MixPosNeg	Total
Dev.	7324	13105	12609	2417	35455
Test	857	1525	1304	255	3941

TABLE 4 – Proportion des données de test et d'entraînement de la tâche 2

positif, 8.7% *négatif*, 33.1% *neutre*, 6.5% *mixposneg*).
Le tableau 5 présente les résultats obtenus dans la tâche 2. Nous constatons que la combinaison est profitable à notre tâche de classification. Si on prend les performances des systèmes individuellement le GRU s'avère le plus efficace.

Type de réseau	Précision	F1-mesure
CNN	0.605	0.754
CNN 2	0.610	0.757
CNN 3	0.639	0.780
BLSTM	0.623	0.768
GRU	0.641	0.781
LSTM	0.632	0.774
Combinaison 1	0.610	0.757
Combinaison 2	**0.657**	**0.793**

TABLE 5 – Précision et F1-mesures des données de test de la tâche 2

Les différents graphiques de la figure 2 mettent en avant le fait que les systèmes détectent avec une meilleure précision les tweets de polarité négative ou neutre (respectivement 67% et 77%). Les tweets de polarité mixte (*mixposneg*) sont mal interprétés (19% de vrais positifs). Cet écart peut s'expliquer par la faible proportion de tweets de ce type dans les données d'entraînement (6.8%). On remarquera également pour ce type de tweet que le label le plus souvent attribué est *négatif* (42%). Les tweets *positifs* sont souvent confondus comme étant des tweets de type *neutres*.

Ces premières observations faites, on pourra s'intéresser à la comparaison des erreurs des différents systèmes. En effet, certains modèles comme le LSTM détectent mieux les tweets *positifs* (60%), le CNN 1 parvient mieux à identifier les tweets *mixposneg* (23%). Ces différences d'interprétation des modèles pouvant aboutir à une meilleure classification, nous avons conduit d'autres expériences consistant à les combiner. Une moyenne des prédictions nous a permis d'améliorer la précision de 1.6%.

Il faut noter que les données contiennent des erreurs d'annotation, qui peuvent fausser l'évaluation des systèmes de classification. Par exemple, les tweets "Ya intérêt que le bus soit à l'heure parce que en retard le premier jour c'est moyen" et "Si vs avez le bac svp ne pensez pas à vous suicider sur la ligne du RER D merci" sont considérés comme *mixposneg* dans la classification de référence, et *négatifs* par notre système.

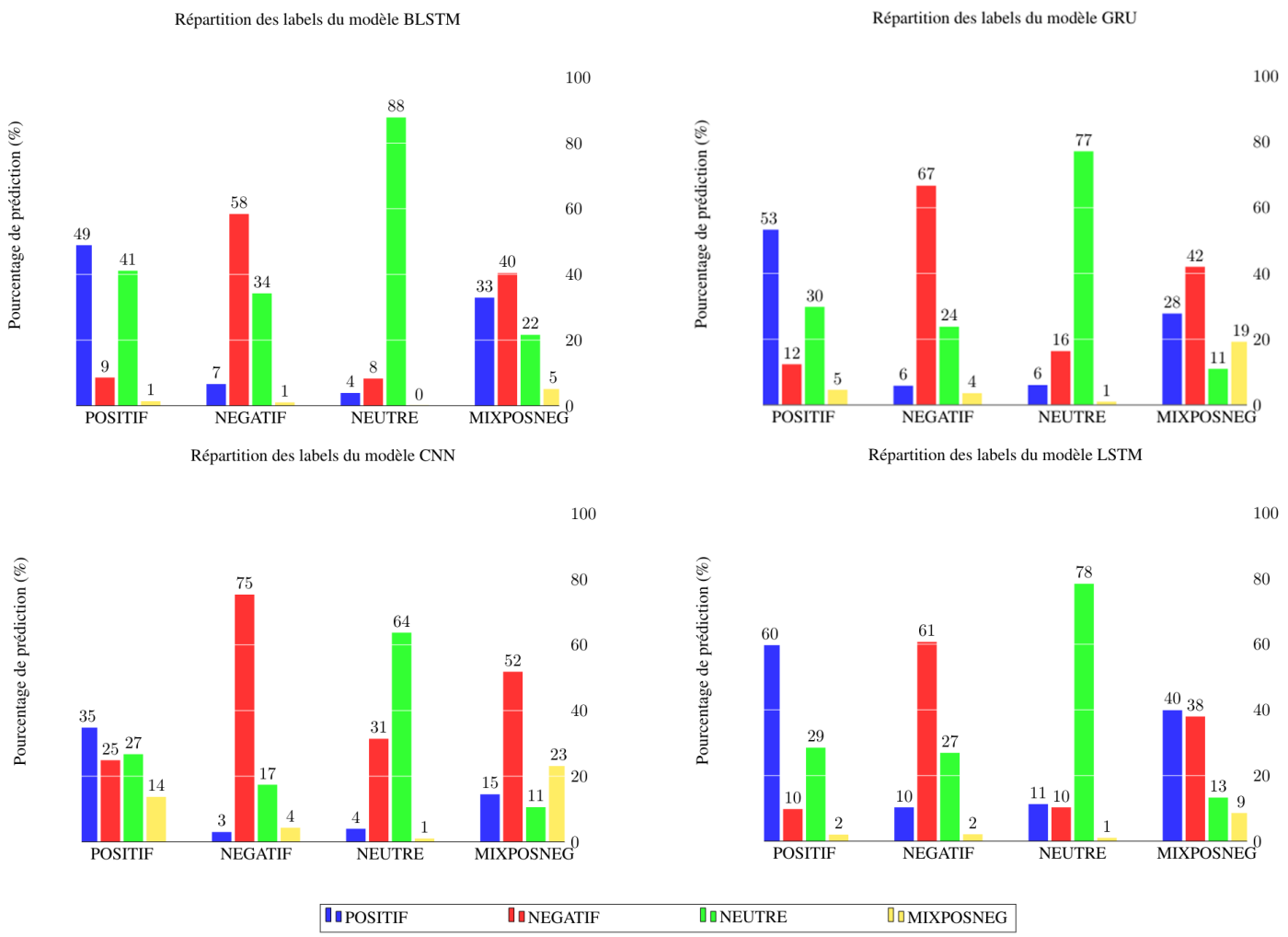

FIGURE 2 – Répartition des prédictions par label de référence des différents modèles de la tâche 2

4 Conclusion

Dans cet article, nous proposons différentes approches basées sur les réseaux de neurones profonds pour classifier les tweets selon leur sujet (*transport* ou *inconnu*) et leur polarité. Pour construire nos plongements de mots, nous avons récolté près d'1 million de tweets traitant de sujets relatifs aux transports. Pour les deux tâches, les résultats obtenus selon l'architecture employée ont des tendances similaires.

Références

BAZIOTIS C., PELEKIS N. & DOULKERIDIS C. (2017). Datastories at semeval-2017 task 6 : Siamese LSTM with attention for humorous text comparison. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada*.

BOJANOWSKI P., GRAVE E., JOULIN A. & MIKOLOV T. (2016). Enriching word vectors with subword information. *CoRR*, **abs/1607.04606**.

CHO K., VAN MERRIENBOER B., GÜLÇEHRE Ç., BAHDANAU D., BOUGARES F., SCHWENK H. & BENGIO Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical

machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,*, p. 1724–1734.

DAS S. & CHEN M. (2001). Yahoo ! for amazon : Extracting market sentiment from stock message boards. In *In Asia Pacific Finance Association Annual Conf. (APFA)*.

DERIU J., GONZENBACH M., UZDILLI F., LUCCHI A., LUCA V. D. & JAGGI M. (2016). Swisscheese at semeval-2016 task 4 : Sentiment classification using an ensemble of convolutional neural networks with distant supervision. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT, USA*.

GÉRON A. (2017). *Hands on Machine Learning with scikit-learn and Tensorflow*. O'Reilly Media.

HOCHREITER S. & SCHMIDHUBER J. (1997). Long short-term memory. *Neural Computation*, **9**(8), 1735–1780.

MIKOLOV T., SUTSKEVER I., CHEN K., CORRADO G. & DEAN J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, **abs/1310.4546**.

PAROUBEK P., GROUIN C., BELLOT P., CLAVEAU V., ESHKOL-TARAVELLA I., FRAISSE A., JACKIEWICZ A., KAROUI J., MONCEAUX L. & TORRES-MORENO J.-M. (2018). Deft2018 : recherche d'information et analyse de sentiments dans des tweets concernant les transports en île de france. In *Actes de DEFT*, Rennes, France.

SCHUSTER M. & PALIWAL K. K. (1997). Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*.

ZHANG Y. & WALLACE B. C. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, **abs/1510.03820**.