LIA @ DEFT'2017 : Multi-view Ensemble of Convolutional Neural Network

Mickael Rouvier et Pierre-Michel Bousquet
University of Avignon
prenom.nom@univ-avignon.fr

RÉSUMÉ

Ce papier décrit le système développé au LIA pour la campagne d'évaluation DEFT 2017. Le but de cette campagne d'évaluation est d'identifier l'opinion exprimée par son auteur sur des tweets. Plusieurs systèmes ont été développés, basés sur des Réseaux de Neurones Convolutionnels (CNN), sur lesquels nous avons fait varier les paramètres et initialisé l'entrée des CNNs avec différents jeux d'*embeddings*. Le système final consiste à fusionner au niveau des scores l'ensemble des CNNs. Le système développé durant la campagne d'évaluation DEFT 2017 a été classé 1^{er} sur les 3 tâches.

ABSTRACT

LIA @ DEFT'2017 : Multi-view Ensemble of Convolutional Neural Network

This paper describes the system developed at LIA for the DEFT-2017 evaluation campaign. The goal of this evaluation campaign was to identify opinion expressed by its author on tweets. An ensemble of Convolutional Neural Network (CNN) was developped, where we varied the network parameters and initialized the input of the CNNs with different sets of word embeddings. The final system is a score-fusion. The system is ranked 1^{th} at DEFT-2017.

MOTS-CLÉS : réseaux de neurones convolutionnels, word embedding, sentiment embedding.

KEYWORDS: convolutional neural network, word embedding, sentiment embedding.

1 Introduction

Cette treizième édition du Défi Fouille de Texte (DEFT) (Benamara *et al.*, 2017) était consacrée à l'analyse d'opinion et du langage figuratif. Le but étant d'identifier l'opinion exprimée par son auteur sur des tweets en français. Pour cette nouvelle édition, trois tâches d'analyse d'opinion ont été proposées :

— **Tâche 1 - Classification des tweets non figuratifs selon leur polarité** : Cette tâche consiste à classer le tweet non figuratif selon l'opinion/sentiment/émotion exprimé(e) par son auteur, en : objectif, positif, négatif ou mixte (si le tweet contient à la fois des opinions positives et négatives).
— **Tâche 2 - Identification du langage figuratif** : Cette tâche consiste à identifier si un tweet contient du langage figuratif ou non. On s'intéresse ici à trois types de langages figuratifs : l'ironie, le sarcasme et l'humour.
— **Tâche 3 - Classification des tweets figuratifs et non figuratifs selon leur polarité** : Étant donné un tweet utilisant du langage figuratif (ironie ou sarcasme, en excluant les tweets humoristiques) ou non, cette tâche consiste à le classer, selon l'opinion/sentiment/émotion exprimé(e) par son auteur en : objectif, positif, négatif ou mixte (si le tweet contient à la fois des opinions positives et des opinions négatives).

Le problème d'analyse d'opinion est souvent modélisé comme un problème de classification qui consiste à extraire des paramètres sur le tweet puis de les fournir à un classifieur. De récents travaux ont montré que les réseaux de neurones convolutionnels (CNN) utilisant une représentation vectorielle des mots comme entrée du classifieur sont bien adaptés aux problèmes de classification de phrase et ont permis d'obtenir des résultats similaires à l'état de l'art pour l'analyse d'opinion (Kim, 2014; Kalchbrenner *et al.*, 2014; Dos Santos & Gatti, 2014; Severyn & Moschitti, 2015). Nous utilisons comme représentation vectorielle de mots les *word embeddings*.

L'approche proposée pour cette campagne d'évaluation consiste à apprendre différents CNNs, dans lesquels nous avons fait varier les paramètres et utilisé différents jeux d'embeddings. Chaque jeu d'embeddings permet de modéliser le tweet d'un point de vue différent. Les différents CNNs entraînés sont ensuite combinés au niveau des scores.

L'équipe Langage du Laboratoire Informatique d'Avignon (LIA) a participé à cette édition et a terminé 1^{er} sur l'ensemble des trois tâches.

Notre contribution est la suivante :

— Un des problèmes avec les *word embeddings* est que les antonymes d'un mot sont placés au même endroit. Nous proposons de nouvelles méthodes afin de capturer la polarité des mots dans les *word embeddings*.
— Une analyse des erreurs de notre système a été faite sur la tâche 1 dans le but de comprendre et améliorer notre système.
— Le code source de notre système, les modèles entraînés pour l'évaluation ainsi que les corpus collectés pour créer les embeddings, sont disponibles à la communauté dans l'espoir d'aider les futures recherches [1].

Nous décrivons dans cet article les techniques et les méthodes automatiques utilisées pour ce défi. La section 2 présente le corpus ainsi que les métriques utilisées lors du défi DEFT 2017. Nous présentons

1. http ://gitlia.univ-avignon.fr/rouvierm/deft-2017

ensuite les étapes de pré-traitement dans la section 3. Dans la section 4, nous présentons les réseaux de neurones convolutionnels et les *word embeddings* dans la section 5. Les résultats apparaissent dans la section 6. Enfin, une discussion et des perspectives sur l'analyse d'opinion est faite dans la section 7.

2 Description de la tâche

2.1 Corpus

Les organisateurs ont mis à la disposition des participants un corpus d'apprentissage composé de 3.906, 5.853 et 5.118 tweets annotés respectivement pour les tâches 1, 2 et 3, et un corpus de test composé de 976, 1.464 et 1.281 tweets respectivement pour les tâches 1, 2 et 3. Le Tableau 1 donne la distribution des tweets par catégorie et par tâche.

TABLE 1 – Distribution des annotations par catégorie et par tâche pour chaque corpus.

Tâche	**Catégorie**	**Apprentissage**	**Test**
T1	objectif	1.643 (42.06%)	411 (42.11%)
	positif	494 (12.65%)	123 (12.60%)
	négatif	1.268 (32.46%)	318 (32.58%)
	mixte	501 (12.83%)	124 (12.70%)
T2	figuratif	1.947 (33.26%)	488 (33.33%)
	non-figuratif	3.906 (66.74%)	976 (66.67%)
T3	objectif	1.718 (33.57%)	430 (33.57%)
	positif	504 (9.85%)	125 (9.76%)
	négatif	2.263 (44.22%)	568 (44.34%)
	mixte	633 (12.37%)	158 (12.33%)

Afin de tester nos méthodes, de régler leurs paramètres et de pallier le phénomène de sur-apprentissage, nous avons décidé de découper le corpus d'apprentissage en 4 sous-ensembles approximativement de la même taille. La procédure d'apprentissage a été la suivante : 3 des 4 sous-ensembles sont concaténés pour produire un corpus d'entraînement et le troisième est utilisé comme corpus de développement. La procédure est effectuée quatre fois afin que chacun des sous-ensembles du corpus d'apprentissage soit utilisé une fois.

2.2 Métrique

Les différents systèmes sont évalués en terme de macro-fmesure, calculée comme suit :

$$rappel_i = \frac{vrai\_positif_i}{vrai\_positif_i + faux\_negatif_i} \quad (1)$$

$$precision_i = \frac{vrai\_positif_i}{vrai\_positif_i + faux\_positif_i} \tag{2}$$

$$fmesure_i = \frac{2 * rappel_i * precision_i}{rappel_i + precision_i} \tag{3}$$

$$macro\_fmesure = \frac{1}{N} \sum_{i=1}^{N} fmesure_i \tag{4}$$

où N est le nombre de classes.

3 Pré-traitements

Une étape de pré-traitement est appliquée aux tweets :

- **Encodage des caractères** : Tous les tweets sont encodés au format UTF-8.
- **Encodage des balises HTML** : Certains caractères ont des significations spéciales en HTML, et doivent être remplacés par des entités HTML (comme par exemple : <, >...).
- **Minuscule** : Tous les caractères sont convertis en minuscule.
- **Rallongement** : Le rallongement des caractères consiste à répéter plusieurs fois un caractère dans un mot. C'est une méthode souvent utilisée sur le web pour insister sur un fait. Ce rallongement est souvent corrélé à un sentiment. Si un caractère est répété plus de trois fois, il sera réduit à trois caractères.
- **Tokenization** : La tokenization réalise le découpage d'une phrase en unités pré-lexicales. Cette tokenization est basée sur le toolkit macaon (maca_tokenize) (Nasr *et al.*, 2011). Il repose sur une grammaire régulière qui définit un ensemble de types d'atomes. Un analyseur lexical détecte les séquences de caractères (en fonction de la grammaire) et leur associe un type. Nous avons rajouté les atomes pour la détection des smileys, hashtags et noms d'utilisateurs (atome spécifique aux tweets).
- **Ponctuation** : Nous supprimons ici tous les caractères de ponctuation.

4 Réseaux de neurones convolutionnels

Ces dernières années, les réseaux de neurones profonds ont obtenu des résultats intéressants dans plusieurs domaines : image, parole... Les réseaux de neurones convolutionnels (CNN) représentent une des méthodes les plus utilisées dans le traitement de l'image (LeCun *et al.*, 1995). De récents travaux ont montré que les CNN sont également bien adaptés aux problèmes de classification de phrase et peuvent produire des résultats comparables à ceux de l'état de l'art (Kim, 2014). La différence entre le CNN appliqué au traitement de l'image et au langage naturel tient dans le format d'entrée. Ainsi dans le traitement de l'image, les entrées sont codées dans une matrice 2D ou 3D, tandis qu'en classification de phrase, chaque entrée est une séquence de mots de taille variable. Chaque entrée w est représentée par un vecteur de dimension-n (*word embedding*) e_w de taille constante. Tous les vecteurs sont ainsi concaténés en respectant leur ordre.

L'architecture du CNN est proche de l'approche proposée dans (Kim, 2014). L'entrée du réseau correspond aux mots d'un tweet tokenisé. La 1^{ere} couche consiste à transformer chaque mot en un vecteur de nombres réels. La couche suivante consiste à appliquer plusieurs matrices de convolution puis une couche d'agrégation (*max-pooling*). Le réseau se termine par une couche cachée et une couche de sortie.

Les paramètres des modèles sont choisis afin de maximiser les performances du corpus de développement : la taille des filtres de convolution, le nombre de convolutions. Nous utilisons les fonctions d'activations ReLU et un max-pooling. Une couche cachée est de taille 128.

Pour ce réseau, un dropout de 0.3 est utilisé (30% des neurones sont désactivés à chaque itération (Nitish Srivastava & Salakhutdinov, 2014)). L'algorithme de back-propagation utilisé pour l'entraînement est l'Adadelta. Dans nos expériences, nous avons observé que les poids d'initialisation des couches de convolution peuvent conduire à une forte variation en terme de performance. Par conséquent, nous entraînons 20 modèles et sélectionnons celui qui a obtenu les meilleurs résultats (macro-fmesure) sur le corpus de développement.

5 Word embedding

La première couche d'un CNN consiste à transformer un mot en un vecteur de nombres réels. Cette transformation est apprise durant la phase d'apprentissage ; mais étant donné la faible quantité de tweets annotés à notre disposition, la transformation ne sera pas assez généralisable et robuste. Nous proposons donc d'initialiser la première couche du CNN en utilisant la transformation obtenue via les *word embeddings*.

Les *words embeddings* sont une approche issue de la sémantique distributionnelle qui consiste à représenter un mot comme un vecteur de nombres réels en fonction de son contexte. Une telle représentation a des propriétés de regroupement intéressantes, car il regroupe les mots qui sont sémantiquement et syntaxiquement similaires (Mikolov *et al.*, 2013). Par exemple, les mots "café" et "thé" seront des vecteurs très proches. Le but est d'utiliser ces paramètres en entrée d'un classifieur. De récents travaux ont montré qu'intégrer des informations sémantiques dans les systèmes d'analyse d'opinion permet d'améliorer les performances mais aussi de réduire la taille des corpus d'apprentissage (Collobert *et al.*, 2011).

Malheureusement, les *word embeddings* ont tendance à placer les antonymes au même endroit dans l'espace (cf. Tableau 2). Par exemple, "confiant" et "méfiant" sont utilisés dans des contextes similaires, et seront des vecteurs très proches. Afin de contourner ce problème, nous proposons d'utiliser des *word embeddings* entraînés suivant différentes formes et ainsi de capturer dans la représentation vectorielle de mots les différentes polarités.

Pour la campagne d'évaluation DEFT 2017, nous explorons différentes approches pour intégrer dans la représentation vectorielle des mots leur polarité. Nous proposons d'apprendre 4 représentations :

— **Lexical embeddings :** Ces embeddings sont obtenus avec le modèle skipgram classique de (Mikolov *et al.*, 2013). La représentation est créée en utilisant la couche cachée d'un réseau de neurones linéaire pour prédire le contexte d'un mot. Pour un contexte donné $w_{i-2} \dots w_{i+2}$, l'entrée du modèle est w_i et la sortie est w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}. Cette méthode extrait une représentation qui couvre à la fois des informations syntaxiques et sémantiques (dans une

certaine mesure).

— **Sentiment embeddings (multitask-learning) :** Un des problèmes avec l'approche skipgram (lexical embeddings) est que le modèle ignore la polarité des mots (les mots "confiant" et "méfiant" sont des vecteurs proches). Dans (Tang *et al.*, 2014), les auteurs proposent de contourner ce problème afin que les informations de polarité soient encodées dans la représentation. Les auteurs proposent de créer un réseau de neurones qui prédit deux tâches : le contexte du mot et la polarité de la phrase. Comme il est vraiment coûteux d'étiqueter manuellement les tweets selon leur polarité, les auteurs proposent d'utiliser les tweets qui contiennent des émoticônes et de relier les mots de cette phrase à l'étiquette de polarité donnée par l'émoticône. Par exemple, si s est la polarité de la phrase où le contexte $w_{i-2} \dots w_{i+2}$ est extrait, l'entrée du modèle est w_i et il doit prédire $(w_{i-2}, s), (w_{i-1}, s), (w_{i+1}, s), (w_{i+2}, s)$.

— **Sentiment embeddings (distant-supervision) :** *Distant-supervision* est une autre solution permettant d'intégrer la polarité des sentiments dans les mots. Un CNN est entraîné sur un corpus sélectionné par des émoticônes positives et négatives. Les émoticônes positives et négatives sont utilisées comme label pour le tweet. Durant l'entraînement, le CNN va automatiquement mettre à jour les *word embeddings* afin de capturer la polarité de sentiment des mots. Les *Word embeddings* mis à jour peuvent être utilisés comme une nouvelle représentation (Go *et al.*, 2009; Severyn & Moschitti, 2015).

— **Sentiment embeddings (negative-sampling) :** Le *negative sampling* est l'approche utilisée pour apprendre les *word embedding*. L'objectif est de produire des représentations vectorielles w et c de dimension d de tout mot w et de tout contexte c. Le critère d'apprentissage consiste à maximiser une fonction du produit $w\dot{c}$ pour les paires (w, c) rencontrées dans le corpus et de minimiser cette même fonction pour des exemples négatifs. Le "negative sampling" consiste à produire les exemples négatifs en tirant aléatoirement, pour chaque paire (w, c), des paires (w, nc). Nous proposons de ne pas tirer les mots nc de manière aléatoire, mais de sélectionner à l'aide d'un dictionnaire de polarité les mots ayant une polarité inverse. Par exemple, pour le mot "bon" nous sélectionnons les mots "mauvais", "terrible", etc...

Pour apprendre les *word embeddings*, nous avons créé un corpus non-annoté de tweets de sentiment en français. Ces tweets ont été récupérés sur la plateforme Twitter en effectuant des recherches avec des mots-clefs porteurs d'émotion, sentiment ou opinion. Ces mots-clefs peuvent être des termes (comme par exemple : vexé, annihilé, ridicule...), des hashtags (#good, #like, #mauvais...) ou des smileys (;), :-), :-D). Ce corpus est composé d'environ 49 millions de tweets en français.

TABLE 2 – Les mots proches de *confiant* et *méfiant* selon les différents manières d'apprendre les word embeddings : lexical, sentiment$_{MultiTask}$, sentiment$_{distant}$ et sentiment$_{negative}$

Lexical		**Sentiment**$_{MultiTask}$		**Sentiment**$_{Distant}$		**Sentiment**$_{Negative}$	
confiant	**méfiant**	**confiant**	**méfiant**	**confiant**	**méfiant**	**confiant**	**méfiant**
optimiste	pessimiste	confiante	utopiste	satisfait	tatillon	optimiste	craintif
inquiet	craintif	serein	tolérant	soulagée	impartial	**inquiet**	inquiet
serein	méfiante	**inquiet**	méfiante	confiante	partial	serein	pessimiste
confiante	impulsif	optimiste	pointilleux	enrichissant	pragmatique	confiante	impulsif
prudent	**optimiste**	dépaysé	docile	amélioré	imho	déterminé	agressif
pessimiste	méfiant	sceptique	idéaliste	contant	hargneux	pessimiste	**optimiste**
sceptique	**confiant**	rodé	intrusif	habile	démérité	prudent	suspicieux
déterminé	inquiet	septique	formel	topissime	réticent	**méfiant**	circonspect
méfiant	agressif	dubitatif	téméraire	impatients	intimement	sceptique	méfiante
timoré	borné	déterminé	préoccupé	discipliné	indépendantiste	attentiste	naïf

6 Résultats

6.1 Architecture du système

Nous faisons varier, dans plusieurs CNNs développés, la taille des filtres de convolution ($\{1,2,3\}$, $\{3,4,5\}$, $\{5,6,7\}$), le jeux d'embeddings (*lexical*, *sentiment*$_{multitask}$, *sentiment*$_{distant}$ et *sentiment*$_{negative}$) ainsi que la dimension des embeddings (*100*, *200* et *300*). Pour chaque sous-ensemble du corpus d'apprentissage, nous avons 36 variantes de CNN.

Le système final est une combinaison linéaire des meilleurs variantes de CNN. Pour chaque sous-ensemble du corpus d'apprentissage, nous sélectionnons les N meilleures variantes de CNN qui optimisent la macro-fmesure. La fusion étant une combinaison linéaire des $4 * N$ variantes de CNN où 4 étant le nombre de sous-ensembles du corpus d'apprentissage. Nous notons que dans la combinaison linéaire toutes les variantes de CNN ont le même poids.

Lors de la campagne d'évaluation DEFT 2017, nous avons pu envoyer 3 différents *runs* :

— **run1** : nous sélectionnons pour chaque sous-ensemble la variante de CNN qui obtient la meilleure macro-fmesure (soit une fusion de 4 variantes de CNN).
— **run2** : nous sélectionnons pour chaque sous-ensemble les 2 variantes de CNN qui obtiennent la meilleure macro-fmesure (soit une fusion de 8 variantes de CNN).
— **run3** : nous sélectionnons pour chaque sous-ensemble les 3 variantes de CNN qui obtiennent la meilleure macro-fmesure (soit une fusion de 12 variantes de CNN).

6.2 Tâche 1

La tâche 1 consiste à prédire le sentiment d'un tweet selon 4 polarités : négatif, positif, objectif et mixte. Le Tableau 3 montre les résultats obtenus sur le corpus de test par les meilleures variantes de CNN apprises sur chacun des sous-ensembles du corpus d'apprentissage. On constate ainsi que le système ayant obtenu les meilleurs résultats (61.22% de macro-fmesure) est celui entraîné sur

le *Sous-ensemble*$_2$. Pour rappel, lorsque l'on parle de *Sous-ensemble*$_2$ cela signifie que le CNN est entraîné sur les sous-ensemble 1, 3 et 4 et que le sous-ensemble 2 est utilisé corpus de développement.

TABLE 3 – Résultats obtenus des variantes de CNN sur la tâche 1

Système	Macro_FMesure	FMesure$_{Pos}$	FMesure$_{Neg}$	FMesure$_{Obj}$	FMesure$_{Mix}$
Sous-ensemble$_1$	59.11	60.76	70.28	79.90	25.49
Sous-ensemble$_2$	61.22	68.29	66.27	77.82	31.51
Sous-ensemble$_3$	59.57	58.37	68.32	78.55	33.05
Sous-ensemble$_4$	57.91	61.54	68.85	78.46	22.80

Le Tableau 4 montre les résultats des différents runs envoyés lors de la campagne d'évaluation. On constate que le système *run3* est le système qui obtient les meilleurs résultats (65.00% de macro-fmesure).

TABLE 4 – Résultats obtenus des différents run sur la tâche 1

Système	Macro_FMesure	FMesure$_{Pos}$	FMesure$_{Neg}$	FMesure$_{Obj}$	FMesure$_{Mix}$
Run1	60.23	64.49	69.63	78.92	27.88
Run2	63.44	66.12	71.05	80.82	35.78
Run3	65.00	67.77	70.52	81.16	40.53

6.3 Tâche 2

La tâche 2 consiste à prédire si le tweet contient un message figuratif ou pas. Le Tableau 5 montre les résultats obtenus sur le corpus de test par les meilleures variantes de CNN apprises sur chacun des sous-ensembles du corpus d'apprentissage. On constate ainsi que le système obtenant les meilleurs résultats (76.86% de macro-fmesure) est celui entraîné sur le *Sous-ensemble*$_1$.

TABLE 5 – Résultats obtenus des variantes de CNN sur la tâche 2

Système	Macro_FMesure	FMesure$_{Fig}$	FMesure$_{NonFig}$
Sous-ensemble$_1$	76.86	70.24	83.48
Sous-ensemble$_2$	75.58	66.59	84.57
Sous-ensemble$_3$	76.06	68.22	83.89
Sous-ensemble$_4$	75.86	67.30	84.42

Le Tableau 6 montre les résultats des différents runs envoyés lors de la campagne d'évaluation. On constate que le système *run1* est le système qui obtient les meilleurs résultats (65.00% de macro-fmesure).

TABLE 6 – Résultats obtenus des différents run sur la tâche 2

Système	Macro_FMesure	FMesure$_{Fig}$	FMesure$_{NonFig}$
Run1	78.31	71.05	85.57
Run2	77.39	69.53	85.25
Run3	77.43	69.54	85.32

6.4 Tâche 3

La tâche 3 consiste à prédire le sentiment d'un tweet selon 4 polarités : négatif, positif, objectif et mixte. Le Tableau 7 montre les résultats obtenus sur le corpus de test par les meilleures variantes de CNN apprises sur chacun des sous-ensembles du corpus d'apprentissage. On constate ainsi que le système obtenant les meilleurs résultats (56.90% de macro-fmesure) est celui entraîné sur le *Sous-ensemble*$_1$.

TABLE 7 – Résultats obtenus des variantes de CNN sur la tâche 3

Système	Macro_FMesure	FMesure$_{Pos}$	FMesure$_{Neg}$	FMesure$_{Obj}$	FMesure$_{Mix}$
Sous-ensemble$_1$	56.90	56.03	70.29	73.63	27.64
Sous-ensemble$_2$	56.61	59.59	70.52	74.30	22.04
Sous-ensemble$_3$	54.45	54.85	70.60	74.18	18.18
Sous-ensemble$_4$	55.26	52.13	66.61	74.43	27.88

Le Tableau 8 montre les résultats des différents runs envoyés lors de la campagne d'évaluation. On constate que le système *run3* est le système qui obtient les meilleurs résultats (59.39% de macro-fmesure).

TABLE 8 – Résultats obtenus des différents run sur la tâche 3

Système	Macro_FMesure	FMesure$_{Pos}$	FMesure$_{Neg}$	FMesure$_{Obj}$	FMesure$_{Mix}$
Run1	57.83	62.55	71.67	75.49	21.60
Run2	58.49	61.67	71.77	75.00	25.51
Run3	59.39	63.71	71.28	76.05	26.51

7 Discussion et perspective

Dans cette section, nous allons discuter tout d'abord des limites de l'annotation du corpus et des limites des approches basé sur les CNNs. Puis dans un second temps, nous effectuons une analyse d'erreur du système.

7.1 Limite de l'annotation du corpus

Un des soucis en analyse d'opinion concerne la classification des tweets. En effet, pour certains tweets, cette classification en opinion n'est pas un problème trivial dû à la frontière entre les classes qui restent relativement floues (ce problème est d'autant plus vrai lorsque le nombre de classes d'opinions augmente). Certains tweets appartenant aux classes "objectif" et "mixed" , conséquence de cette limite floue, peuvent aussi appartenir à d'autres classes telles que "négatif" ou "positif". On se rend bien compte qu'apprendre des étiquettes en dur 0/1 n'est pas approprié, et qu'il serait intéressant d'avoir pour chacun des tweets des probabilités correspondant aux différentes classes.

Pour contourner ce problème, une des approches qui nous semble intéressante est l'approche *teacher-student*. L'approche *teacher-student* consiste à entraîner un modèle état de l'art (modèle *teacher*), et ensuite d'entraîner un nouveau modèle (modèle *student*) pour imiter le modèle *teacher*. Le modèle *student* n'est pas entraîné sur les étiquettes d'origines, mais sur les étiquettes prédites par le modèle *teacher*. Remarquablement, un modèle *student* entraîné à prédire les cibles du modèle *teacher* peut être plus précis que le modèle *teacher* entraîné sur les étiquettes d'origines. Cette approche nous semble aussi intéressante pour d'autres raisons :

— Apprendre des étiquettes en dur 0/1 peut être plus difficile qu'apprendre des probabilités conditionnelles extraites d'un modèle *teacher*.
— Si certaines étiquettes du corpus d'apprentissage comportent des erreurs, le modèle *teacher* peut éliminer certaines de ces erreurs rendant ainsi plus facile l'apprentissage du modèle *student*.
— Le modèle *student* peut être vu comme une forme de régularisation du réseau ce qui permet d'éviter le sur-apprentissage.

Le Tableau 9 donne les résultats du modèle *student*. Le système Model$_{student}$ correspond au modèle *student*. On constate que l'approche *student* permet d'obtenir un gain de 3.16 points par rapport au modèle *Baseline*. Nous nous sommes posé la question de savoir si les bonnes performances n'étaient pas liées à la correction des erreurs présentes dans le corpus d'apprentissage. Nous avons corrigé les erreurs d'annotation et ré-entrainé un nouveau système : le système Baseline$_{corr.}$. On observe alors un gain de seulement 1 point, ce qui tend à confirmer que l'approche *teacher-student* corrige les erreurs liées au corpus, mais aussi permet d'éviter une frontière trop stricte entre les différentes classes.

TABLE 9 – Résultats obtenus des différents systèmes sur la tâche 1

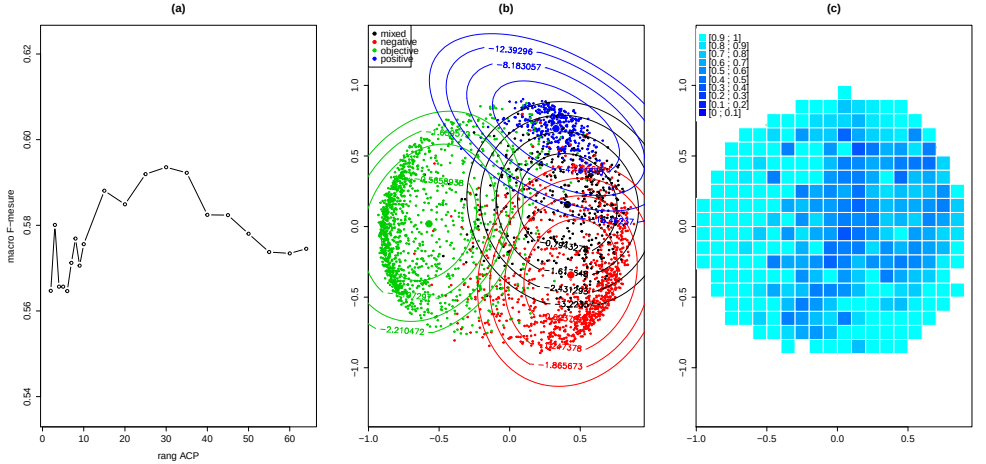
Système	**Macro_FMesure**	**FMesure**$_{Pos}$	**FMesure**$_{Neg}$	**FMesure**$_{Obj}$	**FMesure**$_{Mix}$
Baseline	58.67	60.66	64.38	79.37	30.28
Baseline$_{corr.}$	59.64	63.53	66.18	78.16	30.70
Model$_{Student}$	61.84	64.10	65.72	79.62	37.90

7.2 Limite des approches CNN

Une des hypothèses fortes sur la représentation obtenue par réseaux de neurones est l'indépendance des variables et l'homoscédasticité des classes (égalité des covariances). Cette nouvelle représentation est estimée depuis les données originelles via des transformations linéaires, mais ces hypothèses

ne sont pas nécessairement respectées. Pour pallier ce problème, nous proposons, à partir de la nouvelle représentation obtenue par CNN, d'utiliser la métrique plus sophistiquée de Mahalanobis. Pour chaque tweet nous utilisons la représentation donnée par la couche cachée.

FIGURE 1 – (a) Performance, en termes de macro-fmesure, suivant la dimension des vecteurs réduits par ACP -(b) Vue des vecteurs d'apprentissage réduits par ACP en dimension 2, avec affichage des ellipses de densité gaussienne -(c) Zones de "pureté de classe" de la vue (b).



Les vecteurs obtenus par la couche cachée peuvent être réduits par une Analyse en Composantes Principales (ACP) avant application de la métrique de décision de Mahalanobis. La Figure 1-(a) affiche les performances, en terme de macro-fmesure sur la tâche 1, calculées sur les vecteurs préalablement réduits à des dimensions de 2 à 64 par ACP. Une performance optimale est atteinte pour une dimension de 30 obtenant 59.30% de macro-fmesure.

Si la performance optimale est atteinte pour une dimension 30 (59.3), la réduction à 2 dimensions fournit une performance intéressante (56.4). Ce fait permet d'afficher les représentations vectorielles sur une vue, sans que la validité de celle-ci soit trop limitée. La figure 1-(b) affiche les vecteurs de notre fichier d'apprentissage projetés par ACP en dimension 2. Les couleurs des points indiquent le sentiment. Les distributions gaussiennes des classes sont également représentées par leurs points-moyennes et leurs ellipses de covariance.

Comme le montre la figure, les classes "objective", "positive" et "negative" sont relativement bien séparées, mais la classe "mixed" crée une forte zone d'embrouillement, ce qui était prévisible. Pour mieux mesurer ce constat, la figure 1-(c) affiche les niveaux de "puretés" dans un grillage de l'espace de représentation. Chaque zone carrée est colorée en fonction du niveau de "pureté de classe" des observations qui y sont contenues. Ce niveau est mesuré par le maximum des fréquences d'observations d'une classe dans la zone. Par exemple, une valeur de 0.7 indique que 70% des observations dans cette zone appartiennent à une même classe. La figure montre que les principales zones d'incertitude se situent dans la zone de forte densité des observations de la classe "mixed", qui contribue donc principalement à l'embrouillement entre les classes. Une attention toute particulière devra donc être donnée à cette classe. Une piste intéressante consiste à extraire des paramètres globalement sur la phrase.

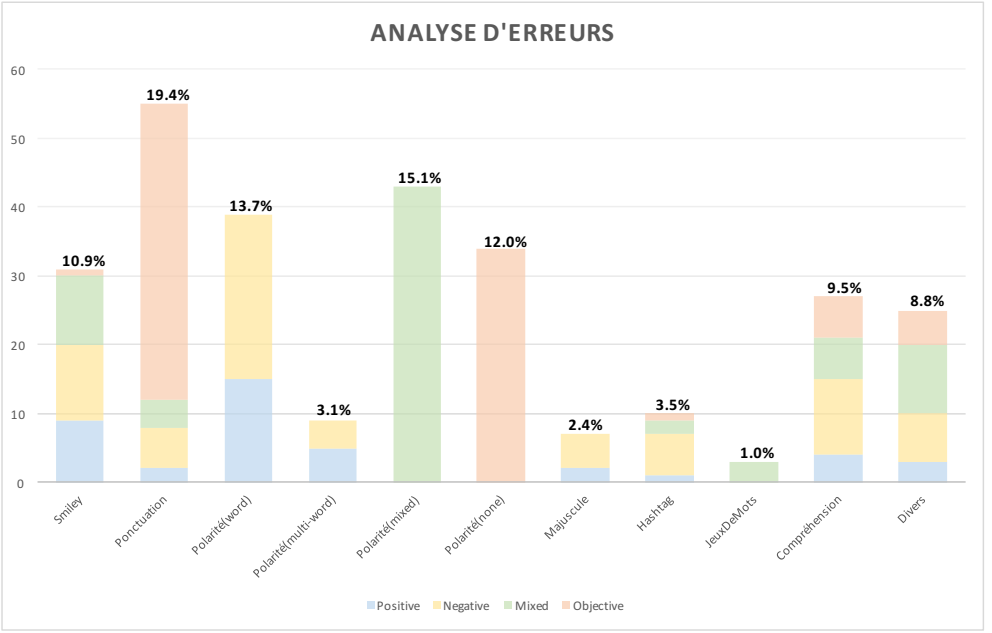
7.3 Analyse des erreurs

Une analyse d'erreurs du système *run3* a été faite sur la tâche 1 dans le but de comprendre les erreurs du système. Nous avons identifié 11 classes qui permettent d'identifier l'opinion d'un tweet :

— **Smiley** : L'opinion du tweet est donnée par les smiley (*:-), :-(*) ou un émoticône
— **Ponctuation** : L'opinion du tweet est donnée par sa ponctuation (citation, point de suspension...)
— **Polarité**$_{word}$: L'opinion du tweet est donnée par un mot (méfiant, confiant...)
— **Polarité**$_{multi-word}$: L'opinion du tweet est donnée par une expression
— **Polarité**$_{mixed}$: L'opinion du tweet est donnée par deux mots
— **Polarité**$_{none}$: Aucune opinion du tweet n'a été donnée
— **Majuscule** : L'opinion du tweet est donnée par un mot ou plusieurs mots en majuscule
— **Hashtag** : L'opinion du tweet est donnée par un hashtag
— **JeuxDeMots** : L'opinion du tweet est donnée par un jeu de mot
— **Compréhension** : détecter l'opinion consiste à avoir une connaissance de l'actualité
— **Divers** : Ne rentre dans aucune des classes précédentes

L'analyse d'erreur est faite sur la tâche 1 en utilisant notre système *run3* (ce système a obtenu les meilleurs résultats lors de la campagne d'évaluation : 65.00% de macro-fmesure). Nous avons classé les tweets parmi les 11 classes sur lesquels le système n'était pas d'accord avec la référence. La Figure 2 permet de voir la répartition des erreurs suivant les différentes classes.

FIGURE 2 – Analyse d'erreurs faites sur la tâche 1 en utilisant le système *run3*.



Nous constatons que "Ponctuation", "Majuscule" et "Smiley" sont des informations très importantes pour la classification en opinion et que ces informations ont été supprimées lors de la tokenization.

De manière générale, la tokenization est une des faiblesses de notre système car elle supprime énormément d'informations utiles pour la classification d'opinion (majuscule, ponctuation, smiley....). Il serait intéressant de pouvoir s'en passer et d'effectuer une classification sans tokenization. Une méthode intéressante consiste à apprendre des CNNs non pas sur les mots, mais sur les lettres du tweet. Ainsi, le CNN appris sur les lettres va s'occuper d'extraire automatiquement sans tokenization les mots, ponctuation, smiley et séparer ainsi l'information utile de l'information inutile.

8 Conclusion

Ce papier décrit la participation du LIA à DEFT 2017. Notre approche consiste à apprendre un ensemble de réseaux de neurones convolutionnels (CNN). Plusieurs systèmes ont été développés, basés sur des CNN, dans lesquels nous avons fait varier les paramètres du réseau et initialisé l'entrée des CNNs avec différents jeux d'*embeddings*. Le système final est une fusion au niveau des scores de ces différentes variantes de CNN. Le système a été classé 1^{er} sur l'ensemble des 3 tâches de la campagne d'évaluation DEFT 2017.

Références

BENAMARA F., GROUIN C., KAROUI J., MORICEAU V. & ROBBA I. (2017). Analyse d'opinion et langage figuratif dans des tweets : présentation et résultats du défi fouille de textes deft2017. *Actes de l'atelier DEFT de la conférence TALN 2017*.

COLLOBERT R., WESTON J., BOTTOU L., KARLEN M., KAVUKCUOGLU K. & KUKSA P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*.

DOS SANTOS C. N. & GATTI M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*.

GO A., BHAYANI R. & HUANG L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*.

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*.

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

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

NASR A., BÉCHET F., REY J.-F., FAVRE B. & LE ROUX J. (2011). Macaon : An nlp tool suite for processing word lattices. In *ACL*.

NITISH SRIVASTAVA, GEOFFREY E HINTON A. K. I. S. & SALAKHUTDINOV R. (2014). Dropout : a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*.

SEVERYN A. & MOSCHITTI A. (2015). Unitn : Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado*.

TANG D., WEI F., YANG N., ZHOU M., LIU T. & QIN B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification.