# Modèles en Caractères pour la Détection de Polarité dans les Tweets

Davide Buscaldi[1] Joseph Le Roux[1] Gaël Lejeune[2]
(1) LIPN, Université Paris XIII, 99 avenue JB Clément, 93430, Villetaneuse
(2) STIH, Sorbonne Université, 28 rue Serpente, 75006, Paris
prenom.nom@lipn.univ-paris13.fr, prenom.nom@sorbonne-universite.fr

## RÉSUMÉ

Dans cet article, nous présentons notre contribution au Défi Fouille de Textes 2018 au travers de trois méthodes originales pour la classification thématique et la détection de polarité dans des *tweets* en français. Nous y avons ajouté un système de vote. Notre première méthode est fondée sur des lexiques (mots et emojis), les n-grammes de caractères et un classificateur à vaste marge (ou *SVM*). tandis que les deux autres sont des méthodes endogènes fondées sur l'extraction de caractéristiques au grain caractères : un modèle à mémoire à court-terme persistante (ou *BiLSTM* pour *Bidirectionnal Long Short-Term Memory*) et perceptron multi-couche d'une part et un modèle de séquences de caractères fermées fréquentes et classificateur *SVM* d'autre part. Le *BiLSTM* a produit de loin les meilleurs résultats puisqu'il a obtenu la première place sur la tâche 1, classification binaire de *tweets* selon qu'ils traitent ou non des transports, et la troisième place sur la tâche 2, classification de la polarité en 4 classes. Ce résultat est d'autant plus intéressant que la méthode proposée est faiblement paramétrique, totalement endogène et qu'elle n'implique aucun pré-traitement.

## ABSTRACT

**Character-level Models for Polarity Detection in Tweets**

We present our contribution to the DEFT 2018 shared task, with three entries based on different methods to perform topic classification and polarity detection for tweets in French, to which we added a voting system. Our first entry is based on lexicons (for words and emojis), character n-grams and a classifier implemented with a support vector machine (*SVM*), while the other two are endogenous methods based on character-level feature extraction : first a long short-memory recurrent neural network (*BiLSTM*) feeding a classifier implementing a multi-layer perceptron, and second a model based on frequent closed character sequences with a SVM. The *BiLSTM* system gave the best results by far. It ranked first on task 1, a binary theme classification task, and third on task 2, a four-class polarity classification task. This result is very encouraging as this method has very few priors, is completely endogenous, and does not require any specific preprocessing.

MOTS-CLÉS : Analyse en Caractères, Bi-LSTM, n-grammes de caractères, Détection de Polarité, Analyse de Tweets.

KEYWORDS: Character-level Models, Bi-LSTM, character n-grams, Polarity detection, Tweet Analysis.

 

# 1 Introduction

L'analyse automatique des *tweets* est un domaine très actif du Traitement Automatique des Langues (TAL), notamment sur l'aspect étude de la polarité et détection des émotions. Ceci n'est pas très étonnant dans la mesure où les *tweets*, et plus généralement les microblogs, constituent un moyen de jauger l'opinion individuelle d'une grande quantité de personnes. Il existe donc un grand nombre d'applications qui pourraient profiter de systèmes performants pour cette tâche. D'un autre côté, la classification binaire de *tweets* selon qu'ils relèvent ou non d'une thématique particulière est une tâche connexe à la détection des émotions mais souvent délaissée, et placée en amont de la chaîne de traitement de TAL. Elle est traditionnellement implémentée par filtrage sur des chaînes de caractères ou des expressions rationnelles qui font référence à des termes du thème visé. C'est regrettable car connaître la polarité d'une émotion est intéressant mais ça l'est plus encore si l'on sait à quoi le *tweet* fait référence.

D'un côté la tâche 1 consiste donc à décider si un *tweet* traite ou non des transports et de l'autre côté la tâche 2 pour laquelle, étant donné un *tweet* traitant des transports, il faut détecter la polarité : Neutre, Négatif, Positif ou Mixte. L'article décrivant cette édition du Défi Fouille de Textes propose une description précise du processus de collecte et d'annotation du corpus (Paroubek *et al.*, 2018), nous ne reviendrons donc pas plus ici sur les données d'entrée.

Dans la section 2 nous décrirons les trois méthodes que nous avons mises en place ainsi qu'un système de vote entre les méthodes. Ensuite, nous présenterons dans la section 3 les résultats que nous avons obtenus dans la compétition officielle ainsi que des expériences plus complètes sur nos méthodes. Enfin, nous proposerons quelques conclusions et perspectives sur ce travail dans la section 4.

# 2 Méthodes : trois modèles en caractères et un système de vote

## 2.1 Méthode 1 : Lexiques et caractères

Le premier système est basé sur des SVM avec un noyau gaussien, où les vecteurs comportent un mélange de caractéristiques à niveau de caractères et lexicales, à l'aide de dictionnaires.

Pour les caractères, nous avons choisi d'utiliser, en tant que caractéristiques, tous les n-grammes de caractères (espaces exclues) de taille comprise entre 3 et 6, avec une fréquence minimale de 100 (arbitrairement choisie) dans le corpus d'entraînement. Ce choix est fondé sur l'observation que les n-grammes de caractères permettent de capturer des variations sur un même mot ou acronyme qui sont fréquentes dans le langage des *tweets*, par exemple : *#ratp*, *@ligne7_ratp*, *#merciratp*, *@grouperatp*, etc. Le poids d'un n-gramme est de 0 s'il n'apparaît pas dans le *tweet* ou $s(n) = \sum_{i=1}^{nbOcc} 1 + pos(n_i)/len(t)$ s'il apparaît dans le *tweet*, où $pos(n_i)$ indique la position du premier caractère de l'$i$-ème occurrence du n-gramme $n$ et $len(t)$ est la taille du *tweet* en nombre de caractères.

Nous avons choisi de garder aussi dans le modèle une composante lexicale, avec l'utilisation de dictionnaires de polarité, en particulier les dictionnaires labMT (Dodds *et al.*, 2011) et FEEL (Abdaoui *et al.*, 2017). Les dictionnaires ont été utilisés pour associer deux poids à chaque *tweet* : un poids pour la première moitié du *tweet* et un autre poids pour la deuxième moitié. Les scores des dictionnaires ont été normalisés dans l'intervalle $[-1, +1]$, dans le cas du FEEL cela revient à associer le label *négatif* à $-1$ et *positif* à $+1$. Pour labMT, étant donné que les scores sont dans l'intervalle $(1, 9)$, on

a associé le minimum à $-1$ et le maximum à $+1$ puis on a mis à l'échelle les valeurs intermédiaires. Finalement pour calculer la valeur des caractéristiques pour chaque moitié du *tweet* on garde la somme des scores des mots.

Pour la tâche T2, le système a utilisé les résultats de la tâche T1 pour travailler uniquement sur les *tweets* étiquetés *Tranport*. On a donc créé un modèle de polarité pour détecter si les *tweets* avaient une polarité ou pas. La sortie de ce modèle a été utilisé comme entrée pour un modèle dédié à la polarité positive et un modèle dédié à la polarité négative. Finalement nous avons produit un script pour combiner les résultats produits par chaque modèle. Donc si un *tweet* avait été reconnu comme *Transport*, alors on vérifie s'il est polarisé ou pas. S'il est polarisé, on vérifie s'il est *positif* ou pas, on vérifie s'il est *négatif* ou pas, et s'il est *positif et négatif* en même temps on lui assigne l'étiquette *MIXPOSNEG*.

## 2.2 Méthode 2 : BiLSTM

Notre deuxième système utilise des réseaux de neurones récurrents pour implanter un classificateur, plus spécifiquement les LSTM (Hochreiter & Schmidhuber, 1997) qui sont largement utilisés en traitement automatiques des langues. La classification se fait en trois temps :

1. Le texte est séparé aux espaces. Chaque segment est traité comme une séquence d'octets lue de gauche à droite et de droite à gauche par deux réseaux récurrents *niveau caractère*. Les vecteurs résultats des lectures sont additionnés et servent de représentation du segment, dite compositionnelle. Pour une séquence de caractères $s = c_1 \dots c_m$, on calcule pour chaque position $h_i = LSTM_o(h_{i-1}, e(c_i))$ et $h'_i = LSTM_{o'}(h'_{i+1}, e(c_i))$, où $e$ est la fonction de plongement des caractères vers les vecteurs denses, et $LSTM$ est un raccourci pour une fonction implantant la cellule récurrente des LSTM. La représentation compositonnelle du segment est $c(s) = h_m + h'_1$
2. La séquence de segments est lue à nouveau de gauche à droite et de droite à gauche par de nouveaux réseaux récurrents *niveau mot* qui prennent en entrée pour chaque segment la représentation compositionnelle venant de l'étape précédente à laquelle on ajoute une représentation vectorielle du segment si celui-ci était présent plus de 10 fois dans le corpus d'entraînement. Pour une séquence de segments $p = s_1 \dots s_n$, on calcule $l_i = LSTM_m(l_{i-1}, c(s_i) + e(s_i))$, $l'_i = LSTM_{m'}(l_{i+1}, c(s_i) + e(s_i))$, où $c$ est la représentation compositionnelle donnée ci-dessus et $e$ la fonction de plongement que l'on étend aux segments vus dans l'ensemble d'entraînement. Les états finaux obtenues après lecture dans les deux directions sont sommés et servent de représentation de la phrase d'entrée, $r(p) = l_n + l'_1$.
3. La représentation obtenue sert d'entrée à un perceptron multi-niveau qui effectue la classification finale, aussi bien pour la tâche 1 que pour la tâche 2 : $o(p) = \sigma(O \times \max(0, (W \times r(p) + b)))$ où $\sigma$ est l'opérateur *softmax*, $W$, $O$ des matrices et $b$ un vecteur. On interprète la sortie comme une distribution de probabilité sur les classes de *tweets*.

Cette interprétation probabiliste nous permet de réduire l'apprentissage des paramètres du système (*ie.* les plongements de caractères et de segments fréquents, $O, W, b$, ainsi que les paramètres des 4 cellules LSTM) à la maximisation de la vraisemblance du corpus d'entraînement. On utilise l'algorithme AMSgrad (Reddi *et al.*, 2018) pour calculer la taille du pas lors de la descente de gradient. Pour éviter le sur-apprentissage, nous procédons aux deux ajustements suivants.

— On écarte aléatoirement du corpus d'entraînement 10% des phrases qui sont utilisées comme ensemble de validation, ce qui permet de décider quand les paramètres sont toujours utiles sur

des données inconnues.
— on utilise la technique du *dropout* (Srivastava *et al.*, 2014), sur tous les vecteurs à chaque étage du réseau — sauf la couche finale bien sûr. Durant l'apprentissage les neurones sont aléatoirement remis à zéro avec une probabilité de 0,5.

Les plongements de caractères sont de taille 16, ceux des segments de taille 32, la couche d'entrée du perceptron est de taille 64 et la couche cachée de taille 32. La couche de sortie est de taille 2 pour la tâche 1, et 4 pour la tâche 2. Nous utilisons la bibliothèque DYNET [1] avec les paramètres par défaut.

## 2.3 Méthode 3 : motifs en caractères

Notre troisième système utilise également une approche d'analyse au grain caractère. Plus exactement cette approche se situe à la lisière entre l'algorithmique du texte et la fouille de données. Nous utilisons des motifs (en caractères) fermés et fréquents comme traits pour entraîner un classificateur.

Les propriétés de fermeture et de fréquence sont définis de la façon suivante :

**Fermeture :** le motif ne peut être étendu vers la gauche ou vers la droite sans diminuer son nombre d'occurrences

**Fréquence :** le motif respecte une borne minimale de nombre d'apparitions

Pour calculer des motifs en caractères de manière efficace, en l'occurrence avec une complexité linéaire en la taille des données, nous utilisons ici une implantation en Python de l'algorithme décrit dans (Ukkonen, 2009) exploitant les tableaux de suffixes augmentés décrits dans (Kärkkäinen *et al.*, 2006). (Buscaldi *et al.*, 2017) rappellent que les propriétés de fermeture et de fréquence correspondent en algorithmique du texte aux propriétés de maximalité et de répétition. Du point de vue du TAL, cette technique peut être décrite comme une segmentation *non-supervisée* en ce sens que les règles de découpage ne sont pas pré-définies mais sont calculées en fonction du corpus donné en entrée.

De façon traditionnelle en fouille de données, un défi important est de limiter l'explosion du nombre de motifs, que ce soit pour des raisons calculatoires ou pour des raisons de lisibilité des résultats.Un filtrage classique consiste à appliquer aux motifs deux types de contrainte :

— La contrainte de support par laquelle on définit le nombre minimal ($minsup$) et maximal ($maxsup$) d'objets qui supportent un motif. Ici cela consiste à définir le nombre minimal et maximal de *tweets* dans lequel le motif apparaît.
— La contrainte de longueur par laquelle on définit la longueur minimale ($minlen$) et maximale ($maxlen$) d'un motif. Ici, il s'agit d'une longueur en caractères.

Notre chaîne de traitement est définie comme suit :

1. Calcul des motifs fermés fréquents dans tout le corpus de *tweets* (train+test) ;
2. Filtrage des motifs selon la longueur ;
3. Filtrage des motifs selon le support ;
4. Représentation de chaque *tweet* sous forme d'un vecteur d'effectif des motifs ;
5. Utilisation de l'implantation de SCIKIT du SVM ONE VS REST.

Durant la phase d'entraînement nous avons testé la méthode au travers d'un validation croisée en 10 strates au moyen de la fonction STRATIFIEDKFOLD de SCIKIT [2]. En examinant les résultats, nous avons observé que la qualité des résultats n'augmentait plus au delà d'un seuil de $maxlen$ de

1. https://github.com/clab/dynet
2. http://scikit-learn.org/

5. Nous avons également remarqué, ce qui est conforme à des résultats précédents sur des données comparables (Buscaldi *et al.*, 2017) que l'application de contraintes de support avait assez peu d'influence sur les résultats. Nous avons donc choisi de ne pas chercher à paramétrer finement cette contrainte. Toutefois, il est à noter que la réduction de l'espace de description engendrée par cette contrainte permet de diminuer le coût en calcul.

Les résultats que nous avons soumis pour la phase de test ont été obtenus avec les configurations suivantes :

— Pas de taille minimale ($minlen = 1$) ;
— Pour la tâche 1 $maxlen = 2$ et pour la tâche 2 $maxlen = 3$ ;
— Pas de contrainte de support ;
— Utilisation d'un noyau linéaire.

La configuration ci-dessus s'est avérée la plus efficace sur les données d'entraînement et la plus fiable pour effectuer les calculs dans le temps imparti. L'utilisation d'un noyau radial permettait toutefois d'obtenir de bons résultats y compris en se contentant des motifs de taille 1 ($maxlen = 1$).

### 2.4 Systèmes de vote

Avec notre quatrième run, nous avons tenté de tirer profit des propriétés respectives de nos trois méthodes en élaborant un système de vote. Pour les deux tâches, il s'agit simplement d'un vote majoritaire entre les résultats des trois méthodes. Pour la tâche 2 qui comporte 4 classes, il a fallu gérer les cas d'égalité qui représentaient près de 25% des cas. Nous avons considéré que l'absence d'accord tangible entre les méthodes indiquait que nous avions à faire à des *tweets* sans tendance particulière. Nous avons donc donné l'étiquette *MIXPOSNEG*. Les cas d'indécision étant plutôt fréquents, cela a abouti à une sur-représentation de cette classe qui a nui aux résultats comme nous le montrerons dans la section suivante.

## 3 Résultats

### 3.1 Tâche 1 : classification thématique *Transport* ou *Inconnu*

Cette tâche était relativement facile avec beaucoup de configurations au-delà des 80% de F-mesure. Nos systèmes se sont bien comportés et le BiLSTM a même obtenu la première place. Les résultats officiels figurent dans le tableau 1.

| Run | Précision | Rappel | F-mesure | VP | FP | FN |
|---|---|---|---|---|---|---|
| Run1 (Lexique et caractères) | 0,80463 | 1.0 | 0,89174 | 6289 | 1527 | 0 |
| Run2 (BiLSTM) | 0,831246497 | 1.0 | 0,90785 | 6497 | 1319 | 0 |
| Run3 (motifs en caractères) | 0,77364 | 0,999 | 0,87231 | 6046 | 1769 | 1 |
| Run4 (vote) | 0,824826446 | 0,999 | 0,90394 | 6446 | 1369 | 1 |

TABLE 1 – Résultats officiels de Tweetaneuse sur la tâche 1

Assez logiquement au vu de la facilité de la tâche, nos trois systèmes ont souvent été en accord complet (71,9% des cas précisément). Le système de vote n'a pas fonctionné aussi bien que nous

l'aurions souhaité car nous n'avons pas pu détecter de réelle complémentarité entre les systèmes. Les systèmes 1 et 3 ont rarement eu raison *contre* le BiLSTM (40 cas soit seulement 0,5% des cas). Dans 320 cas (soit 4,1%), il y a eu uniquement un de nos systèmes qui a été correct. Enfin, dans 1049 cas (soit 13,41%), aucun de nos systèmes n'avait trouvé la bonne étiquette, dont 93 reprises pour la catégorie transports, ce qui est assez attendu puisque la classe *Inconnu* s'était avérée la plus difficile à détecter dans la phase d'entraînement.

Derrière ces erreurs nous pouvons les difficultés inhérentes à un tâche d'annotation même binaire. Toutefois, nous avons pu voir des *tweets* dont la classification était vraiment étonnante.

Quelques exemples étiquetés dans la catégorie *Transport* mais classés *Inconnu* par nos 3 systèmes :

- `Remember à la #CDM2014 quand un tracteur est passé dans toutes les rues de Berche pour nous récupérer et chanter la Marseillaise`
- `Malheureusement pour la #fra ce soir les Marines US ne débarqueront pas par la mer pour la libérer contre la #ger ...#EURO2016`
- `@scanlan75018 depuis qu' il a eu le coup il n' arrive plus à accélérer mais comme c' est une finale il à essayer de forcer`

Quelques exemples étiquetés dans la catégorie *Inconnu* mais classés *Transport* par nos 3 systèmes :

- `La prochaine fois que la SNCF me met dans le sens inverse de la marche alors qu' il reste plein de places dans le train je vomis partout.`
- `@LIGNEJ_SNCF #ligne Merci au conducteur qui vient de partir de houilles de m'avoir attendu`
- `Tu pars bcp plus tard le matin et bcp plus tôt le soir. Ben ça change rien c' est l' horreur sur la @Ligne1_RATP Super mois d' août à la #ratp`
- `#duflot2017 veut rouler en bus au poireau bio? enfin la politique me fait rire!! Celui de son mec ne l'est pas?? MDR`
- `En même temps , y' a qu' un bus pour toute la journée , il n' allait quand même pas passer`

En analysant brièvement les *tweets* pour lesquels aucun de nos systèmes n'a trouvé la bonne classe, nous avons pu voir quelques cas intéressants. Les *tweets* mêlant bus et football contenaient souvent une référence que nos systèmes n'avaient pas pu modéliser : *mettre le bus* dans le sens de se ruer en défense et *rester dans le bus* en référence au bus de Knysna lors de la Coupe du Monde 2010. Un autre cas est le *tweet* ou il n'est pas réellement question de transports mais où le transport sert au mieux de toile de fond. Il s'agit par exemple de cas où le twittos indique explicitement qu'il se situe dans les transports sans que cela ait aucun lien avec ce dont il est train de parler. En voici quelques exemples :

- `c pas je marchais sereine pour aller prendre mon bus et je remarque y' a un bouton de ma chemise ouvert , dévoilant tt mon soutif`
- `à l' arrêt de bus g vu mon prof de philo j' étais pas sereine`
- `Jsuis encore à l' arrêt de bus et je pense à ce qui m' attend ce soir niveau révisions ptdr go mourir ===>`
- `Hier dans le métro j' ai vu un mec il avait des mains!! s' il te donne une gifle t' es mal...`

## 3.2 Tâche 2 : classification en termes de polarité

Dans le tableau 2 sont recensés nos résultats avec les métriques officielles. Une fois encore, le modèle BiLSTM s'est avéré le plus compétitif avec une troisième place sur cette tâche. Dans cette tâche, le différentiel entre nos système s'est creusé et nous avions un triple accord dans seulement 1033 cas (soit 26,2% des cas). Dans 1391 cas (35,3%) deux des systèmes ont fourni la bonne réponse, influent positivement les résultats du système de vote. Enfin, pour 622 *tweets* (15,78%) aucun de nos systèmes n'a trouvé la bonne classe.

| Run | Précision | Rappel | F-mesure | VP | FP | FN |
|---|---|---|---|---|---|---|
| Run1 (Lexique et caractères) | 0,47599 | 0,93313 | 0,63041 | 1814 | 1997 | 130 |
| Run2 (BiLSTM) | 0,67699 | 1 | 0,80738 | 2668 | 1273 | 0 |
| Run3 (Motifs en caractères) | 0,57828 | 1 | 0,7328 | 2279 | 1662 | 0 |
| Run4 (vote) | 0,62766 | 0,9996 | 0,77113 | 2473 | 1467 | 1 |

TABLE 2 – Résultats officiels de Tweetaneuse sur la tâche 2

Nous présentons dans le tableau 3 les résultats par classe pour chacune de nos méthodes. Nous indiquons également la macro F-mesure qui permet de mettre un peu en lumière les configurations qui s'accommodent le mieux de la classe minoritaire *MIXPOSNEG*. Cette classe s'est révélée très difficile à prédire en plus de son faible effectif du fait de sa définition forcément plus imprécise. Nous pouvons observer que le BiLSTM a fait la différence sur toutes les classes. Il s'est aussi mieux comporté que les autres dans la détection des *tweets* de la classe *MIXPOSNEG*.

Parmi les erreurs que nos systèmes ont commis, nous avons extrait quelques exemples pour lesquelles la prédiction n'était pas évidente.

Voici une première série de *tweets* pour lesquels nos 3 systèmes ont prédit *NEGATIF* mais où l'annotation était *MIXPOSNEG* :

- `RT @allenlafrance : #PLC veulent se débarrasser des Postes,on va leur suggérer faire de même avec RC,CBC. et gérer les affaires du Pays. htt...`
- `- Le contrôleur lui dit que ça doit être un pb de date et le mec sûr de lui sort ' Ah c' est pas de ma faute ' , il sort donc son billet et dit-`
- `La #SNCF a vraiment un des pire service clients que je connaisse! C est boite est une honte vivement la concurrence`

Quelques exemples de *tweets* pour lesquels nos 3 systèmes ont prédit *POSITIF* mais où l'annotation était *MIXPOSNEG* :

- `jsuis morte laura elle est descendu de son bus juste pour me dire bonjour à arrêt et remonter mdrrrrr`
- `@J_Lutecia @Virginiement Yep alors par contre j' ai cette contrainte de dernier bus qui fait que ça serait cool si on pouvait faire ça tot :3`
- `les gens ils pensent jsuis l' genre à courir après pour les capter PTDRRRR archi drôle , déjà que j' cours pas après mon bus alors vous nvm`

| CLASSE | Précision | Rappel | F-mesure | VP | FP | FN |
|---|---|---|---|---|---|---|
| run1 (Micro F1 : 0,4761, Macro F1 : 0,3717) | | | | | | |
| NEUTRE | **0,7783** | 0,1414 | 0,2393 | 179 | 51 | 1087 |
| POSITIF | 0,5856 | 0,4145 | 0,4855 | 342 | 242 | 483 |
| NEGATIF | 0,5136 | **0,8393** | 0,6373 | 1243 | 1177 | 238 |
| MIXPOSNEG | 0,0882 | 0,2125 | 0,1247 | 51 | 527 | 189 |
| run2 (Micro F1 : 0,677, Macro F1 : 0,5946) | | | | | | |
| NEUTRE | 0,6854 | **0,75** | **0,7162** | 978 | 449 | 326 |
| POSITIF | **0,6254** | **0,6429** | **0,6341** | 551 | 330 | 306 |
| NEGATIF | **0,7306** | 0,7043 | 0,7172 | 1074 | 396 | 451 |
| MIXPOSNEG | **0,3988** | 0,2549 | **0,311** | 65 | 98 | 190 |
| run3 (Micro F1 : 0,5783, Macro F1 : 0,4872) | | | | | | |
| NEUTRE | 0,5741 | 0,7132 | 0,6361 | 930 | 690 | 374 |
| POSITIF | 0,5591 | 0,4527 | 0,5003 | 388 | 306 | 469 |
| NEGATIF | 0,6557 | 0,6007 | 0,627 | 916 | 481 | 609 |
| MIXPOSNEG | 0,1957 | 0,1765 | 0,1856 | 45 | 185 | 210 |
| run4 (Micro F1 : 0,6278, Macro F1 : 0,5475) | | | | | | |
| NEUTRE | 0,7474 | 0,6127 | 0,6734 | 799 | 270 | 505 |
| POSITIF | 0,6973 | 0,5134 | 0,5914 | 440 | 191 | 417 |
| NEGATIF | 0,6957 | 0,7541 | **0,7237** | 1150 | 503 | 375 |
| MIXPOSNEG | 0,1446 | **0,3333** | 0,2017 | 85 | 503 | 170 |

TABLE 3 – Résultats détaillés classe par classe pour chaque run de la tâche 2

# 4 Discussion

Dans cet article, nous avons présenté trois méthodes exploitant le grain caractère pour la classification de *tweets* et la détection de polarité. Ces trois méthodes utilisent de l'apprentissage supervisé. La première méthode combine ressources exogènes, des lexiques, et n-grammes de caractères, la seconde exploite un BiLSTM tandis que la troisième utilise des motifs en caractères fermés fréquents.

Les bons résultats obtenus, en particulier pour le BiLSTM montrent que l'utilisation du grain caractère pour les tâches de TAL est amenée à avoir un impact important. En effet, les méthodes en caractère ont l'avantage de bien se comporter dans des contextes bruités, ce qui est le cas ici puisque les variations de graphie et la distanciation avec les normes syntaxiques sont des caractéristiques majeures des *tweets*. Par ailleurs, ces approches permettent de se passer de pré-traitement ce qui permet de simplifier les chaînes de traitement et par conséquent de réduire les erreurs en cascade(Lejeune *et al.*, 2015). Enfin, ces méthodes permettent de capturer de manière purement endogène des informations sur la structure intra-token (morphèmes) comme sur la structure extra-token (expressions multi-mots par exemple) ce qui peut être particulièrement intéressant dans un contexte impliquant plusieurs langues ou des langues peu dotées en ressources. Nos prototypes sont disponibles en ligne [3] et librement utilisables.

3. https://github.com/rcln/tweetaneuse2018

# Références


ABDAOUI A., AZÉ J., BRINGAY S. & PONCELET P. (2017). FEEL : a French Expanded Emotion Lexicon. *Language Resources and Evaluation*, **51**(3), 833–855.

BUSCALDI D., GREZKA A. & LEJEUNE G. (2017). Tweetaneuse : Fouille de motifs en caractères et plongement lexical à l'assaut du deft 2017. In *Actes du 13e Défi Fouille de Texte*, p. 65–76, Orléans, France : Association pour le Traitement Automatique des Langues.

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.

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

KÄRKKÄINEN J., SANDERS P. & BURKHARDT S. (2006). Linear work suffix array construction. *Journal of the ACM*, p. 918–936.

LEJEUNE G., BRIXTEL R., DOUCET A. & LUCAS N. (2015). Multilingual event extraction for epidemic detection. *Artificial Intelligence in Medicine*. doi : 10.1016/j.artmed.2015.06.005.

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.

REDDI S. J., KALE S. & KUMAR S. (2018). On the convergence of adam and beyond. In *International Conference on Learning Representations*.

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.

UKKONEN E. (2009). Maximal and minimal representations of gapped and non-gapped motifs of a string. *Theoretical Computer Science*, p. 4341–4349.