Concaténation de réseaux de neurones pour la classification de tweets, DEFT2018

Damien Sileo[1, 2,*] Tim Van de Cruys[2,*] Philippe Muller [2] Camille Pradel [1]
(1) Synapse Développement, 5 Rue du Moulin Bayard, 31000 Toulouse
(2) IRIT, Université Paul Sabatier 118 Route de Narbonne 31062 Toulouse
(*) Contributions égales
damien.sileo@synapse-fr.com, camille.pradel@synapse-fr.com,
philippe.muller@irit.fr, tim.van-de-cruys@irit.fr

RÉSUMÉ
Nous présentons le système utilisé par l'équipe Melodi/Synapse Développement dans la compétition DEFT2018 portant sur la classification de thématique ou de sentiments de tweets en français. On propose un système unique pour les deux approches qui combine concaténativement deux méthodes d'embedding et trois modèles de représentation séquence. Le système se classe 1/13 en analyse de sentiments et 4/13 en classification thématique.

ABSTRACT
Concatenation of neural networks for tweets classification, DEFT2018

We present the system used by the Melodi / Synapse Development team in the DEFT2018 competition on the classification of themes or sentiments of tweets in French. We propose a unique system for both approaches that combines concatentively two embedding methods and three sequence representation models. The system ranks 1/13 in sentiment analysis and 4/13 in thematic classification.

MOTS-CLÉS : *Fasttext*, classification, ensemble.

KEYWORDS: *Fasttext*, classification, ensemble.

1 Introduction

La classification de textes est une application importante du traitement des langues, instanciée sur de nombreux aspects : identification de langue, analyse de sentiment, détection de contenu haineux, catégorisation thématique pour n'en citer que quelques uns. Son application à la communication sur les réseaux sociaux tels que Twitter a néanmoins nécessité quelques déclinaisons permettant d'appréhender au mieux la concision et la "liberté" du langage. La constante évolution des techniques de classification complique leur comparaison objective, en particulier sur des domaines différents comme le français.

La campagne DEFT2018 (Paroubek *et al.*, 2018) propose 4 taches liées au TALN sur des tweets en français, dont 2 taches de classification où chaque tweet est associé à une classe y.

1. La classification thématique des tweets : $y \in \{\text{TRANSPORT}, \text{INCONNU}\}$.
2. L'analyse de sentiments des tweets : $y \in \{\text{POSITIF}, \text{NEGATIF}, \text{MIXPOSNEG}, \text{NEUTRE}\}$. MIXPOSNEG concerne les sentiments partagés et NEUTRE les sentiments peu marqués.

On propose d'évaluer des systèmes généraux ayant la même configuration d'hyperparamètres sur les deux taches.

2 Modèle

2.1 Vue d'ensemble

On utilise un système avec deux niveaux de représentations. D'abord les mots sont représentés par deux composantes :

— Des embeddings fixés (représentations issues du modèle *Fasttext*, qui prennent en compte la morphologie des mots) ;
— Des embeddings appris.

Cette approche dite multi-canaux (Kim, 2014) permet d'exploiter à la fois les régularités du corpus d'entraînement non supervisé et des données DEFT dans les représentations de mots.

Puis ces composantes partagent le rôle d'entrée pour trois modèles de représentation de séquences :

— Un réseau de convolution 1D profond, dont l'architecture sera détaillée. Ce modèle permet de détecter des motifs pertinents pour la classification.
— La moyenne de tous les embeddings de mots présents dans la phrase, suivie d'une projection et d'une non-linéarité. Cette composante permet de prendre en compte également tous les mots de la phrase et de représenter un aspect thématique/contextuel plus global.
— Un réseau récurrent (GRU (Chung *et al.*, 2014)) qui est le modèle le plus général des trois, capable de capturer d'autres statistiques de la nature séquentielle des tweets.

Ce système est entraîné 15 fois pour chaque tache avec des initialisations différentes sur 90% des données d'entraînement. Les 8 meilleurs systèmes d'après le score F1 sur les données de validation restantes sont retenues. La prédiction est alors réalisée avec la moyenne des estimations de probabilité par classe fournies par chaque modèle. Par ailleurs la partie fixe des embeddings est issue d'un apprentissage de *Fasttext* à chaque fois différente. Cette stratégie sert à augmenter la variance entre les modèles pour améliorer l'ensemble.

La figure 1 montre l'architecture utilisée. Les nombres entre crochets sont les dimensions.

2.2 *Fasttext*

Le modèle skipgram *Fasttext*(Schmidhuber, 2015) est basé sur le modèle skipgram de *word2vec* (Mikolov *et al.*, 2013), qui consiste à apprendre des représentations de mots pour qu'elles optimisent une tâche de prédiction du contexte des mots. La différence principale est que la représentation h_w d'un mot w ne se résume plus à u_w, la représentation de son symbole. Elle est augmentée de la représentation des n-grammes de caractères contenus dans w, nommés $u_g, g \in \mathcal{G}_w$:

$$h_w = u_w + \sum_{g \in \mathcal{G}_w} u_g \tag{1}$$

$\mathcal{G}_w$ correspond aux n-grammes de w suffisamment fréquents et d'une taille adéquate. La morphologie de w est donc partiellement prise en compte dans h_w, même si l'ordre des grammes est ignoré.

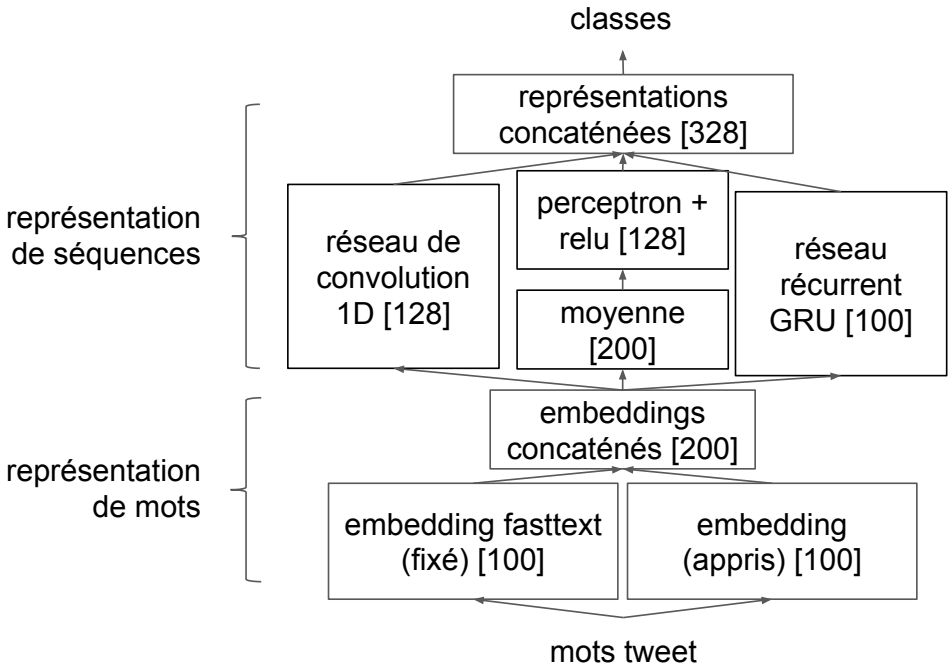

FIGURE 1 – Architecture du système proposé

2.3 Réseau de convolution

Pour l'architecture convolutive, nous nous sommes appuyés sur le modèle décrit par Kim (2014). Dans une couche convolutionnelle, un certain nombre de *feature maps* (filtres) sont appliquées à une fenêtre de mots c (représentés comme des embeddings). Nous appliquons ensuite un *max-over-time pooling* au résultat de la convolution, qui sélectionne la valeur maximale dans une fenêtre particulière en tant que caractéristique correspondant à un filtre particulier.

Il a été démontré que les architectures convolutionnelles profondes améliorent la classification des textes (Conneau *et al.*, 2017) ; nous utilisons un total de trois couches convolutives (128 filtres avec une taille de fenêtre c de 2), suivies de trois étapes de pooling. Dans les deux premières étapes de pooling, nous regroupons deux valeurs ; la dernière étape de pooling est un pooling global sur l'ensemble du contexte, de sorte que nous nous retrouvons avec une valeur unique pour chacun des 128 filtres. La représentation résultante est envoyée à une couche dense (également à 128 valeurs), et une couche finale softmax est utilisée pour la classification.

L'architecture spécifique du modèle et les hyperparamètres ont été choisis en fonction de la performance sur un ensemble de validation.

3 Expériences

3.1 Pré-entraînement des embeddings

Pour apprendre les représentations de mots de *Fasttext*, nous avons utilisé des tweets stockés sur la plateforme *OSIRIM*[1] de l'IRIT qui collecte 1% du flux de Twitter depuis Septembre 2015. À cela s'ajoutent les données de DEFT2018 issues des phases de train et de test. Les tweets sont dé-

1. http://osirim.irit.fr/site/fr/articles/corpus

paramètres	valeur
learning rate	0.02
dimensions	100
context window size	5
epochs	4
min_count	5
negative/positive samples ratio	5
loss	negative sampling
minimum n-gram size	3
maximum n-gram size	6
sampling threshold	10^{-4}

TABLE 1 – Paramètres du modèle *Fasttext*

dupliqués, passés en minuscules, et les liens ou occurrences des *[ASCII012CTRLC]* sont remplacés par des symboles unicodes rares (Å et $\mathbb{U}$) pour que leurs caractères ne polluent pas celles des représentations de *Fasttext*. L'espace précédents les apostrophes dans les données de DEFT2018 est enlevé. L'ensemble résultant totalise $50M$ tweets. Les paramètres utilisés par *Fasttext* sont résumés dans la table 1.

3.2 Méthodologie et autres hyperparamètres

La taille du vocabulaire est fixée aux $50k$ mots les plus fréquents dans les données de DEFT2018 (train et test). Seuls les espaces sont utilisés pour réaliser la tokenization.

Le seul hyperparamètre choisi avec validation croisée rigoureuse est la régularisation L2 du softmax final choisie dans $\{10^p, p \in [[-12, -4]]\}$. 10^{-12} a été retenu.

Un dropout de 0.3 est utilisé après les embeddings, ainsi qu'après les 3 systèmes de représentation de séquences.

Les paramètres sont appris par deux algorithmes d'optimisation utilisés successivement : 2 époques avec Adam (Kingma *et al.*, 2014), avec les paramètres par défaut, puis 1 époque de descente de gradient classique avec un taux d'apprentissage de 10^{-5}. La norme des gradients est seuillée de sorte à ne pas dépasser 3.

4 Evaluation

Le tableau 4 présente les résultats fournis par le système d'évaluation pour nos différents runs, puis des statistiques sur les meilleurs systèmes de chaque équipe à titre de comparaison. L'entraînement joint consiste à entraîner les modèles sur T1 et T2 en même temps, et en inférence à considérer seulement les catégories de la tache considérée. Le GRU a été enlevé dans les deux premiers runs. Le résultat de cette ablation est intéressante puisque les opérations du GRU n'étant pas parallélisables, il ralentit significativement les calculs. Pourtant, il n'améliore pas les résultats sur la tache 1 et seulement ponctuellement sur la tache 2. L'apport de l'entraînement joint n'est stable ni en changeant

	T1	T2
1- Entrainements séparés sans GRU	**0.90371**	0.82165
2- Entrainement joint sans GRU	0.90232	0.80413
3- Entrainements séparés	0.90155	0.81918
4- Entrainement joint	0.88367	**0.82288**
médiane des concurrents	0.895025	0.77304
meilleur ou meilleur suivant	**0.90739**	0.81313

TABLE 2 – Résultats des runs. La médiane est celle des meilleurs runs de chaque équipe, et on reporte également le score de la meilleure équipe pour la tache T1 et deuxième pour la tache T2.

la tache ni en enlevant/ajoutant le GRU.

Parmi les systèmes résultants, 3 auraient gagné la compétition restreint à la tache 2, dont le système 1, également classé 4/13 sur la tache 1 et qui semble être le plus robuste.

5 Conclusion

Nous avons décrit un système général présenté à la compétition DEFT2018. En restant général, il serait intéressant d'évaluer l'apport de méthodes d'apprentissage non supervisées traitant l'ordre des mots (Kiros *et al.*, 2015; Nie *et al.*, 2017). Une optimisation plus rigoureuse et exhaustive des hyperparamètres, ou des techniques de maximisation du score F1 (Chase Lipton *et al.*, 2014) pourraient également améliorer les résultats. Enfin, la détection de tweets liés au transport pourrait très sûrement bénéficier de représentations adaptées de mots rares, spécifiques ou de sigles, même si Fasttext traite partiellement ce problème.

Références

CHASE LIPTON Z., ELKAN C. & NARAYANASWAMY B. (2014). Thresholding Classifiers to Maximize F1 Score. *ArXiv e-prints*.

CHUNG J., GULCEHRE C., CHO K. & BENGIO Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv*, p. 1–9.

CONNEAU A., SCHWENK H., BARRAULT L. & LECUN Y. (2017). Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 1, Long Papers*, p. 1107–1116, Valencia, Spain : Association for Computational Linguistics.

KIM Y. (2014). Convolutional Neural Networks for Sentence Classification.

KINGMA D., REZENDE D. & WELLING M. (2014). Semi-supervised Learning with Deep Generative Models. In *arXiv preprint arXiv : ...*, p. 1–9 : Nips.

KIROS R., ZHU Y., SALAKHUTDINOV R. R., ZEMEL R., URTASUN R., TORRALBA A. & FIDLER S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*, p. 3294–3302.

MIKOLOV T., CHEN K., CORRADO G. & DEAN J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Nips*, p. 1–9.

NIE A., BENNETT E. D. & GOODMAN N. D. (2017). DisSent : Sentence Representation Learning from Explicit Discourse Relations.

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.

SCHMIDHUBER J. (2015). On Learning to Think : Algorithmic Information Theory for Novel Combinations of Reinforcement Learning Controllers and Recurrent Neural World Models. *arXiv :1604.00289v1[cs.AI]*, p. 1–55.