Participation de l'IRISA à DeFT2017 : systèmes de classification de complexité croissante

Vincent Claveau [1, 3] Christian Raymond[2, 3]
(1) CNRS
(2) INSA Rennes, Rennes , France
(3) IRISA/INRIA, Rennes, France
vincent.claveau@irisa.fr, christian.raymond@irisa.fr

RÉSUMÉ

Cet article décrit la participation de l'équipe LinkMedia de l'IRISA à DeFT 2017. Notre équipe a participé aux 3 tâches : classification des tweets non figuratifs selon leur polarité (tâche 1), l'identification du langage figuratif (tâche 2) et la classification des tweets figuratifs et non figuratifs selon leur polarité (tâche 3). Pour ces trois tâches, nous adoptons une démarche d'apprentissage artificiel. Plus précisément, nous explorons l'intérêt de trois méthodes de complexité croissante : i) les k plus proches voisins issues de la recherche d'information, ii) le boosting d'arbres de décision, et iii) les réseaux neuronaux récurrents. Nos approches n'exploitent aucune ressource externe riche (lexiques, corpus annotés) et sont uniquement fondées sur le contenu textuel des tweets (et d'autres tweets pour la dernière approche). Cela nous permet d'évaluer l'intérêt de chacune de ces méthodes, mais aussi des représentations qu'elles exploitent, à savoir les sacs-de-mots pour la première, les n-grams pour la deuxième et le plongement de mots (*word embedding*) pour les réseaux neuronaux.

ABSTRACT

IRISA at DeFT2017 : classification systems of increasing complexity

This paper describes the participation the LinkMedia team from IRISA at DeFT2017. We carried out the three proposed tasks : classification of non-figurative tweets according to their polarity (task 1), identification figurative langage (task 2), classification of figurative and non-figurative tweets according to their polarity (task 3). For these three tasks, we adopt a standard supervised machine learning framework and explore the use of three methods of increasing complexity : i) k-nearest neighbors with information retrieval based techniques, ii) boosting of decision trees, iii) recurrent neural networks. Our approaches do not exploit rich external knowledge (lexicons, annotated corpora) and are chiefly based on the tweet content (and some other tweets for our latest approach). It allows us to evaluate the precise interest of each of our approach and the data representation that they use : bag-of-words for the first one, n-grams for the second and word embedding for the latest.

MOTS-CLÉS : K-plus-proches voisins, boosting, arbres de décision, réseau de neurones récurrents, plongement de mots.

KEYWORDS: K-nearest neighbors, boosting, decision trees, recurrent neural networks, word embedding.

1 Introduction

Cet article détaille les systèmes que l'IRISA a développés dans le cadre de sa participation à DeFT2017 (Benamara *et al.*, 2017). Notre équipe a participé aux 3 tâches proposées par les organisateurs :

— tâche 1 : classification des tweets non figuratifs selon leur polarité (représenté par l'une des quatre classes suivantes : *objective, positive, negative, mixed*) ;
— tâche 2 : l'identification du langage figuratif (classes *figurative, nonfigurative*) ;
— tâche 3 : classification des tweets figuratifs et non figuratifs selon leur polarité (mêmes classes que la tâche 1).

Pour ces trois tâches, nous avons adopté une démarche d'apprentissage artificiel supervisé tout à fait classique. Les données d'entraînement founies nous ont permis d'inférer des classifieurs ensuite utilisés pour la prédiction sur les données de test.

Nous nous sommes attachés à explorer l'intérêt de trois méthodes de classification de complexité croissante :

1. les k plus proches voisins, avec des techniques issues de la recherche d'information ;
2. le boosting d'arbres de décision ;
3. les réseaux neuronaux récurrents.

Nos approches n'exploitent aucune ressource externe riche (lexiques d'opinion, corpus annotés) et sont uniquement fondées sur le contenu textuel des tweets (et d'autres tweets pour la dernière approche). Cela nous permet d'évaluer l'intérêt de chacune de ces méthodes, mais aussi des représentations qu'elles exploitent, à savoir les sacs-de-mots pour la première, les n-grams pour la deuxième et le plongement de mots (*word embedding*) pour les réseaux neuronaux.

La suite de l'article présente successivement ces trois approches au sein des sections 2, 3, et 4. Les résultats accompagnés de quelques commentaires sont ensuite présentés en section 5 avant de conclure l'article en évoquant quelques perspectives à ce travail.

2 K plus proches voisins

La classification par k-plus proches voisins est une technique particulièrement simple de fonctionnement. Son principe est d'assigner à un objet la classe majoritaire parmi ses voisins. Dans sa version la plus simple, il n'y a donc pas de phase d'apprentissage à proprement parler, le saut inductif est assuré par le choix d'une métrique adaptée pour trouver les plus proches voisins.

Malgré sa simplicité, cette approche a déjà montré de bons résultats sur des tâches de classification de texte, et même de tweets comme nous l'avons montré lors de la campagne DeFT 2015 (Vukotic *et al.*, 2015a).

2.1 Choix de la fonction de similarité

Le cœur de l'approche est donc la définition d'une fonction de distance ou de similarité pour trouver les voisins les plus proches du tweet à classer. Pour ce faire, nous utilisons des approches simples issues de la recherche d'information (RI) : le tweet à classer est considéré comme une requête et un système de RI permet de trouver les tweets (dont le label est connu) les plus proches. Les tweets sont

ainsi décrits comme des sacs de mots, chaque mot est pondéré (voir ci-dessous), et une fonction de similarité est utilisée pour calculer un score de pertinence (RSV pour *Relevance Status Value*) entre le tweet requête et un autre tweet.

Dans le cadre de notre participation, nous avons utilisé la fonction de similarité Okapi-BM25 (Robertson *et al.*, 1998). Le tweet à classer, considéré comme une requête, noté q, contient les mots t_i. Le score RSV est calculé pour chacun des tweets d de l'ensemble d'entraînement, de la manière suivante :

$$\begin{aligned} RSV_{BM25}(q,d) &= \sum_{t_i} qTF(t_i,q) * TF_{BM25}(t_i,d) * IDF_{BM25}(t_i) \\ &= \sum_{t_i} \frac{(k_3+1)*tf(t_i,q)}{k_3+tf(t_i,q)} * \frac{tf(t_i,d)*(k_1+1)}{tf(t_i,d)+k_1*(1-b+b*dl(d)/dl_{avg})} * \log\frac{N-df(t_i)+0.5}{df(t_i)+0.5} \end{aligned} \quad (1)$$

avec df la fréquence documentaire d'un mot (nombre de tweets contenant ce mot), dl la longueur du tweet, dl_{avg} la longueur moyenne des tweets dans l'ensemble d'entraînement, et $k_1 = 1$, $b = 0.75$, $k_3 = 1000$ sont des constantes.

2.2 Apprentissage

Le seul paramètre libre du kppv est le nombre de voisins considérés (k) pour le vote. Dans notre cas, il a été choisi par validation-croisée à 10 plis sur les données d'entraînement. Pour les tâches 1 et 3, il est de $k = 15$, pour la tâche 2, de $k = 5$.

3 Boosting

3.1 Prétraitements

Le texte des tweets a été peu retravaillé : une étape basique de tokenisation a été appliquée afin de décoller la ponctuation. Chaque mot du tweet a été représenté sous 3 formes :

1. le mot lui même,
2. le mot en minuscule,
3. sa catégorie d'appartenance à une des classes (parti politique, media, journaliste, personnage politique, hastag, url, @)

Des N-grammes sont ensuites extraits depuis ces trois représentations par le logiciel bonzaiboost (Laurent *et al.*, 2014) qui les utilise pour produire un modèle de boosting d'arbres de décision. Différentes profondeur d'arbres, de valeur de N-grammes ont été testées par validation croisée à 10 plis, aucune combinaison particulière ne semble faire mieux que les paramètres les plus basiques : unigramme et arbres de décision à 2 feuilles dont les résultats sur la tâche 1,2 et 3 sont respectivement visibles dans les tableaux 1,2 et 3. Il est à noter que les optimisations ont été faites en fonction des micro-mesures et non des macro-mesures et ce sont celles là qui sont reportées (les macro-mesures sont quand à elles facilement calculables avec les informations présentées).

Nous avons tenté de représenter la classe « mixed » par une double appartenance aux classes « positives » et « négative » (comme pour notre approche réseau de neurones ; voir section 4.3) plutôt qu'une classe « mixed » explicite, mais ceci n'a pas fonctionné.

Label	Hypo.	Ref.	TP	FP+TN	Précision	Rappel	F1	%erreur
mixed	55	124	14	110	25.45	11.29	15.64	88.71
negative	372	318	213	159	57.26	66.98	61.74	50.00
objective	448	411	336	112	75.00	81.75	78.23	27.25
positive	101	123	62	61	61.39	50.41	55.36	49.59
All	976	976	625	351	64.04	64.04	64.04	35.96

TABLE 1 – Résultats sur la tâche 1 avec un modèle de boosting unigramme et arbre à 2 feuilles et 2000 itérations

Label	Hypo.	Ref.	TP	FP+TN	Précision	Rappel	F1	%erreur
figurative	411	488	291	197	70.80	59.63	64.74	40.37
nonfigurative	1053	976	856	197	81.29	87.70	84.38	20.18
All	1464	1464	1147	317	78.35	78.35	78.35	21.65

TABLE 2 – Résultats sur la tâche 2 avec un modèle de boosting unigramme et arbre à 2 feuilles et 1000 itérations

3.2 Analyse

La similitude de résultat entre la tâche 1 et 3, qui ont été traitées de la même manière, amène une première conclusion : il semble inutile de déduire le style figuratif ou non du tweet pour en déduire la polarité de l'opinion exprimée et il ne semble pas plus compliqué de prédire l'opinion d'un tweet utilisant un style figuratif qu'un d'un autre. La détection quant à elle du style figuratif n'est pas très aisée avec 40% d'erreurs (voir tableau 2), le tableau (4) des règles caractéristiques des tweets figuratif n'est pas très informatifs, il permet tout de même de faire ressortir des personnages qui ont été « moqués » dans l'actualité comme Jean-François Copé ou Morandini.

En regardant les règles caractéristiques des polarités d'opinions exprimées produites par le modèle de boosting, notamment sur la tâche 3 (voir tableau 5), on voit que les tweets « positifs » sont notamment agrémentés d'émoticônes ou de smileys positifs ; les tweets « négatifs » se repèrent facilement lorsqu'ils contiennent des mots négatifs, mais globalement et en dehors de ça, il semble difficile de trouver des éléments fortement généralisateur d'une opinion.

Label	Hypo.	Ref.	TP	FP+TN	Précision	Rappel	F1	%erreur
mixed	54	158	13	145	24.07	8.23	12.26	91.77
negative	675	568	437	238	64.74	76.94	70.31	41.90
objective	455	430	320	135	70.33	74.42	72.32	31.40
positive	97	125	54	71	55.67	43.20	48.65	56.80
All	1281	1281	824	457	64.32	64.32	64.32	35.68

TABLE 3 – Résultats sur la tâche 3 avec un modèle de boosting unigramme et arbre à 2 feuilles et 600 itérations

vote « figuratif »		vote« non figuratif »	
Corée	1.484	#reinesdushopping	1.653
grève	1.455	@olivierminne	1.613
dopage	1.354	Lavrilleux	1.555
#mariagepourtous	1.258	#x1f60d;	1.553
people	1.254	#BurgerKing	1.512
#Morandini	1.088	and	1.469
#EstimeUnPrixCommeCope	1.084	fr	1.465
bravo	1.035	#erdogan	1.392
venir	1.029	Manif	1.386
angleterre	1.010	❤?	1.329
retard	0.953	😘	1.329
années	0.920	#meilleurpatissier	1.314
@jf_cope	0.917	allé	1.314
Algérie	0.914	Ankara	1.313
TF1	0.904	l'Etat	1.288
merci	0.892	Lagarde	1.287
#Copé	0.892	tue	1.286

TABLE 4 – 17 Règles (observées au moins 10 fois dans l'apprentissage) les plus fortement pondérées par l'algorithme de boosting sur la tâche 2, accompagnées de leurs poids, pour les styles de tweet figuratif et non figuratif

opinion positive		opinion négative		opinion objective	
#x1f60d;	2.061	pauvre	1.620	Turquie :	1.373
❤?	1.888	nul	1.329	groupe	1.334
👌	1.641	sarko	1.222	l'attentat	1.322
😍	1.640	plein	1.087	Brexit :	1.320
😘	1.577	gvt	1.034	Henri	1.142
👍	1.335	grosse	0.981	hausse	1.111
@VictorArtus	1.333	connait	0.957	allé	1.077
magnifique	1.308	pauvre	0.926	Lagarde	1.042
bravo	1.226	honte	0.910	longueur>=121.5	1.040
:D	1.219	démocratie	0.880	forces	1.033
LA	1.195	confiance	0.873	annonce	1.027
Besancenot	1.120	ah	0.865	#Ankara	1.020

TABLE 5 – 13 Règles (observées au moins 10 fois dans l'apprentissage) les plus fortement pondérées par l'algorithme de boosting sur la tâche 3 accompagnées de leurs poids, pour les 3 catégories d'opinion les plus représentées

4 Réseau de neurones

L'utilisation de réseaux de neurones dans les tâches liées aux sciences de données est devenue très commune. Le TAL n'échappe pas à cette tendance, et de nombreuses tâches de classification, d'annotation ou de génération de textes sont traitées avec succès par ces méthodes. Il était donc intéressant pour nous de comparer ces approches, aux précédentes.

4.1 Description des données

Nous avons adopté un cadre standard en terme de description des données. Les tweets sont décrits mot par mot, et chaque mot est représenté par un vecteur issu d'un plongement (*embedding*) construit sur les principes d'analyse distributionnelle. Devant le peu de données disponibles, il nous a paru plus approprié d'apprendre ce plongement sur des données externes et de ne pas l'adapter aux tâches. Un plongement a été appris sur une collection de 300k tweets en français liés aux élections présidentielles collectés à l'occasion du projet NexGenTV [1]. Nous avons classiquement utilisés l'outil Word2Vec (Mikolov *et al.*, 2013) dans son implémentation proposée par Gensim (Řehůřek & Sojka, 2010) avec pour paramètres principaux une dimension = 300 avec une fenêtre = 2. Il est à noter que des expériences, non rapportées ici, montrent la faible influence de ces paramètres sur les résultats finaux.

4.2 Architecture

En terme d'architecture, cependant, nous avons voulu tirer au mieux parti des tâches telles qu'elles étaient définies pour limiter l'effet du manque de données. En effet, les réseaux de neurones permettent d'apprendre des classifieurs très complexes, dans le sens où ils comportent de très nombreux paramètres apprenables. Ils nécessitent donc un grand nombre de données pour que ces paramètres soient estimés correctement.

Les ensembles d'entraînement fournis pour chacune des trois tâches ne nous semblant pas suffisants pour apprendre indépendamment des réseaux spécialisés, nous avons adopté une architecture unique permettant de traiter les trois tâches.

Tout d'abord, nous ne distinguons pas les tâches 1 et 3 : on cherche donc à classer les tweets en *objective, positive, negative* ou *mixed*, sans faire de distinction entre ceux issus de la tâche 1 et ceux issus de la tâche 3. Les exemples de ces deux tâches sont donc fusionnés. Intuitivement, cela signifie que l'on suppose que le statut de tweet *figuratif* ou non (ceux de la tâche 1 sont uniquement non figuratifs) est soit peu important pour classer parmi les quatre classes, soit, s'il est important, dérivable par notre réseau. Dans les expériences menées, cela se justifie expérimentalement puisque cette approche nous permet de gagner en moyenne 3 points de micro-F1.

Ensuite, notre réseau a deux couches de sorties, celle pour la tâche 1 et 3, et celle pour la tâche 2, mais les premières couches du réseau sont communes à toutes les tâches. L'intuition est que ces premières couches, décrivant les données, sont indépendantes de la tâche. Les couches suivantes, séparées selon les tâches, permettent de spécialiser cette description selon la tâche.

Enfin, la couche de sortie de la tâche 2 est réemployée en entrée comme description supplémentaire

1. http ://www.nexgentv.fr/

pour la partie du réseau dédiées aux tâches 1/3. Cela permet de prendre en compte le statut *figuratif* ou non du tweet, en supposant que cette prédiction est correcte.

Le reste de l'architecture est standard. Une couche d'entrée prend les plongements des mots composant le tweet examiné. Une couche GRU (Chung *et al.*, 2014) bidirectionnelle permet de prendre en compte l'aspect séquentiel de ces mots. Ensuite deux branches pour nos deux tâches se composent chacune d'une autre couche GRU, d'une couche cachée dense et d'une couche de sortie (voir sous-section suivante). Pour éviter le sur-apprentissage, des couches de *DropOut* (Hinton *et al.*, 2012) sont intégrées entre chacune des couches vues précédemment.

4.3 Sortie et optimisation

La représentation des classes pour les couches de sortie a été la suivante. Pour la tâche 2, un neurone de sortie indique si le tweet est figuratif (1) ou non (0) ; pour les tâches 1 et 3, un neurone indique la dimension positive et un autre la dimension négative ; ainsi, la classe *objective* est codée 00, la classe *mixed*, 11.

Pour la phase d'apprentissage, nous avons utilisé une fonction de perte binaire classique pour ces problèmes avec sorties binaires indépendantes. Nous y avons tout de même apporté une modification pour gérer le fait d'utiliser les données de toutes les tâches conjointement : nous n'avons pas la vérité terrain de tous les tweets d'entraînement pour toutes les tâches (par exemple, on ne connaît pas le statut *figuratif* ou non de certains tweets du jeu d'entraînement de la tâche 3). Notre modification de la fonction de perte permet d'utiliser une sortie 'inconnu' qui ne met donc pas à jour les poids de la partie du réseau concernée.

L'apprentissage se fait en 30 itérations, ce qui correspond à un plateau en terme de performance sur un ensemble de validation tiré aléatoirement parmi les données d'entraînement. La taille du *batch* est fixée à 64, mais dans nos expériences, ce paramètre influence la vitesse de convergence mais peu les résultats finaux.

5 Expériences

5.1 Résultats par tâche

Le tableau 6 rassemblent les scores tels que fournis par les organisateurs de DeFT 2017.

Plusieurs points importants sont à noter. Tout d'abord, nous n'avons pas cherché à optimiser selon la macro-F1. Comme nous l'avons déjà expliqué et démontré (Vukotic *et al.*, 2015a), cette mesure présente deux problèmes lorsque les classes sont déséquilibrées (c'est notamment très largement le cas pour les tâches 1 et 3). D'une part elle correspond rarement à un besoin réaliste, où l'on préfère un système performant en moyenne sur tous les objets à classer (et donc performant en terme de micro-F1). D'autre part, et c'est plus problématique, elle mène à des scores très fluctuants et les différences entre systèmes, même larges, sont rarement statistiquement significatives.

Ensuite, les trois classifieurs obtiennent des résultats assez proches, avec un léger avantage au réseau de neurones. Cela peut s'expliquer par la représentation des données, permettant de mieux prendre en compte des variations orthographiques ou lexicales sans effet sur le sens et la classe finale. Le

tâche	méthode	macro-F1	micro-F1
1	boosting	0.512	0.644
	kppv	0.511	0.658
	DNN	0.514	**0.676**
2	boosting	0.740	0.778
	kppv	0.641	0.741
	DNN	0.744	**0.782**
3	boosting	0.508	0.643
	kppv	0.508	0.651
	DNN	0.517	**0.687**

TABLE 6 – Résultats de notre participation à la campagne de deft2017 pour les 3 tâches et nos trois systèmes

kppv semble légèrement plus performant que le boosting d'arbres sur les tâche 1 et 3 ; à l'inverse, le boosting obtient de meilleurs résultats sur la tâche 2.

Par ailleurs, on note sans surprise que les performances obtenues aux tâches 1 et 3 sont similaires, alors que la tâche 1 se voulait plus facile que la tâche 3 puisque ne portant que sur des tweets non figuratifs.

Enfin, il est intéressant de mettre ces différences de performances en regard des coûts calculatoires d'apprentissage. Ceux-ci sont quasi nuls pour le kppv, puisqu'aucun paramètre n'est optimisé. Bonzaiboost repose sur une phase d'apprentissage plus importante, dépendant principalement de la profondeur des arbres inférés et du nombre d'itérations du boosting. L'apprentissage du réseau de neurones, dont les paramètres sont très nombreux, est largement plus coûteux mais reste raisonnable étant donné le faible nombre d'exemples à notre disposition pour les trois tâches. Sur un ordinateur portable, sans carte GPU, cela représente environ 30 minutes pour l'architecture présentée.

5.2 Quantité de données

Il est intéressant d'examiner le comportement de nos trois systèmes en fonction du nombre de données d'apprentissage. La figure 1 présente l'évolution de la micro-F1 sur la tâche 3 en fonction du nombre de données d'apprentissage utilisées.

On constate que le kppv atteint très rapidement un plateau ; il ne nécessite que peu de données d'entraînement pour atteindre ses performances maximales. Cela s'explique aisément par l'absence de paramètres à apprendre, qui le rend donc peu sensible au sur- et sous-apprentissage. La courbe de notre technique de boosting révèle au contraire une dépendance plus marquée à la quantité de données d'apprentissage. On constate par ailleurs qu'elle semblerait un meilleur choix que le kppv si plus de données avait été fournies pour la tâche. Enfin, le réseau de neurones obtient rapidement des performances supérieures aux deux autres approches. Cela peut sembler surprenant étant donné le grand nombre de paramètres à estimer avec ce type d'approches, mais cela s'explique par la présence de nombreux mécanismes tendant à favoriser la généralisation tout en limitant le sur-apprentissage (par exemple, les mécanismes de DropOut). Et là encore, l'évolution de la courbe semble indiquer que des données supplémentaires permettraient d'améliorer les résultats.

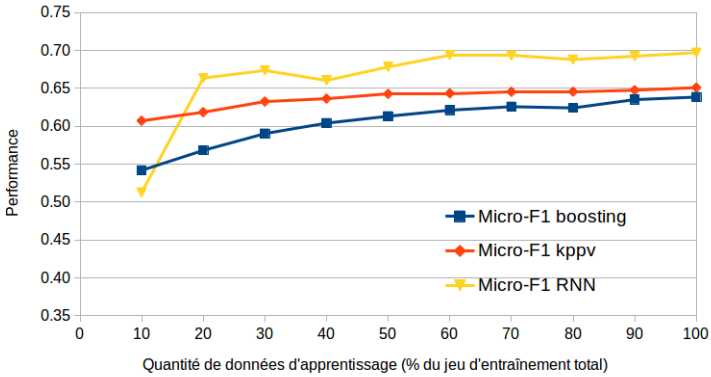

FIGURE 1 – Évolution de la performance (micro-F1) sur la tâche 3 en fonction de la quantité de données d'apprentissage utilisée (en % des données d'entraînement initiales) pour les trois systèmes proposés. Pour chaque point, l'apprentissage a été réalisé dix fois et les résultats moyennés.

6 Conclusion

Parmi, les trois systèmes présentés, boosting, kppv, et réseaux de neurones, et ce malgré des résultats globalement moyens et du même ordre, les réseaux de neurones produisent les meilleures prédictions. Ceci est vraisemblablement du à la représentation vectorielle issue de plongements (*embedding*) des tweets, cette représentation permettant probablement de mieux généraliser et d'être plus résistante (Vukotic *et al.*, 2015b) au bruit produit par des annotations « subjectives ». Nous espérons vérifier cette hypothèse en utilisant ces représentations vectorielles au sein de nos autres classifieurs (BonzaiBoost et kppv), ou d'autres plus adaptées à ces représentations numériques (SVM par exemple) et ainsi mesurer ce qui relève de la représentation et du classifieur.

Ces résultats permettent tout de même de tirer des conclusions plus globales sur la tâche. Notamment, la distinction faite entre la tâche 1 et la tâche 3 ne paraît pas pertinente puisque celles-ci peuvent se traiter de la même façon, avec les mêmes données d'apprentissage et des niveaux de performance similaires.

Remerciements

Ce travail a été partiellement financé grâce à une aide de l'État attribuée au labex COMIN LABS et gérée par l'Agence Nationale de la Recherche au titre du programme « Investissements d'avenir » portant la référence ANR-10-LABX-07-01.

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. In *In*

CHUNG J., GÜLÇEHRE Ç., CHO K. & BENGIO Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, **abs/1412.3555**.

HINTON G. E., SRIVASTAVA N., KRIZHEVSKY A., SUTSKEVER I. & SALAKHUTDINOV R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv :1207.0580*.

LAURENT A., CAMELIN N. & RAYMOND C. (2014). Boosting bonsai trees for efficient features combination : application to speaker role identification. In *InterSpeech*, Singapour.

MIKOLOV T., CHEN K., CORRADO G. & DEAN J. (2013). Efficient Estimation of Word Representations in Vector Space. In *International Conference on Learning Representations*.

ŘEHŮŘEK R. & SOJKA P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, p. 45–50, Valletta, Malta : ELRA. `http://is.muni.cz/publication/884893/en`.

ROBERTSON S. E., WALKER S. & HANCOCK-BEAULIEU M. (1998). Okapi at TREC-7 : Automatic Ad Hoc, Filtering, VLC and Interactive. In *Proc. of the 7th Text Retrieval Conference, TREC-7*, p. 199–210.

VUKOTIC V., CLAVEAU V. & RAYMOND C. (2015a). IRISA at DeFT 2015 : Supervised and Unsupervised Methods in Sentiment Analysis. In *DeFT, Défi Fouille de Texte, joint à la conférence TALN 2015*, Actes de l'atelier DeFT, Défi Fouille de Texte, joint à la conférence TALN 2015, Caen, France.

VUKOTIC V., RAYMOND C. & GRAVIER G. (2015b). Is it time to switch to Word Embedding and Recurrent Neural Networks for Spoken Language Understanding ? In *InterSpeech*, Dresde, Germany.