Stratégie Multitâche pour la Classification Multiclasse

Houssam Akhmouch, Hamza Bouanani, Gaël Dias, Jose G. Moreno

Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles
Lille, France, 28 juin au 2 juillet 2021

Stratégie Multitâche pour la Classification Multiclasse

Houssam Akhmouch[1, 2] Hamza Bouanani[2] Gaël Dias[1] José G. Moreno[3]
(1) Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France
(2) Crédit Agricole Brie Picardie, 77100 Meaux, France
(3) Université de Toulouse, IRIT UMR 5505 CNRS, 31000 Toulouse, France
first.last@unicaen.fr, first.last@ca-briepicardie.fr, first.last@irit.fr

RÉSUMÉ

Nous proposons une idée originale pour exploiter les relations entre les classes dans les problèmes multiclasses. Nous définissons deux architectures multitâches de type *one-vs-rest* qui combinent des ensembles de classifieurs appris dans une configuration multitâche en utilisant des réseaux de neurones. Les expériences menées sur six jeux de données pour la classification des sentiments, des émotions, des thématiques et des relations lexico-sémantiques montrent que nos architectures améliorent constamment les performances par rapport aux stratégies de l'état de l'art de type *one-vs-rest* et concurrencent fortement les autres stratégies multiclasses.

ABSTRACT

A Multitask Strategy for Multiclass Classification

We propose an original idea to exploit the relations between classes in multiclass problems. We define two multitask one-vs-rest architectures that combine sets of classifiers learned in a multitask way using neural networks. Experiments over six gold standard data sets for sentiment, emotion, topic and lexico-semantic relations classification show that our architectures steadily improve performance compared to state-of-the-art one-vs-rest strategies, and strongly compete with other multiclass strategies.

MOTS-CLÉS : classification multitâche, classification multiclasse, classification de textes, classification de relations lexico-sémantiques.

KEYWORDS: multitask classification, multiclass classification, text classification, classification of lexico-semantic relations.

1 Introduction

Alors que la classification binaire traite des problèmes à deux classes (par exemple, spam vs. non spam), la classification multiclasse implique l'attribution de plus de deux étiquettes (par exemple, positif, neutre ou négatif). Un large éventail de problèmes multiclasses existe dans le monde réel, tels que l'analyse des sentiments (Balikas *et al.*, 2017), la classification des genres de nouvelles (Wang *et al.*, 2018), la détection des émotions (Ye *et al.*, 2020), pour n'en citer que quelques-uns. Si de nombreux efforts ont été déployés pour (1) affiner des modèles d'apprentissage pour des problèmes spécifiques de classification textuelle (Teng *et al.*, 2016; Zhou & Li, 2020), ou (2) concevoir des architectures (neuronales) spécifiques à la classification de texte (Conneau *et al.*, 2017; Yao *et al.*, 2019; Zhang & Zhang, 2020), moins de travaux se sont attelés à proposer des modèles génériques qui peuvent traiter des problèmes multiclasses indépendamment de la tâche et de l'objet à classer.

Dans ce cadre, trois approches principales ont été proposées pour traiter les problèmes multiclasses. Premièrement, ils peuvent être résolus en étendant des techniques de classification binaire ; directement

pour les arbres de décision, les réseaux bayésiens ou les K plus proches voisins, et avec quelques modifications pour les réseaux de neurones (Ou & Murphey, 2007) et les machines à vecteurs de support (Crammer & Singer, 2001). Deuxièmement, des stratégies ont été développées qui reposent sur une décomposition du problème en un ensemble de sous-problèmes binaires, dont les résultats sont combinés pour déterminer la solution multiclasse finale. Dans ce cadre, deux stratégies principales ont été largement utilisées, à savoir la stratégie *one-vs-rest* et la stratégie *one-vs-one*. Selon (Hsu & Lin, 2002), les deux stratégies offrent souvent des performances similaires, tandis que des résultats contrastés sont observés par (Pawara *et al.*, 2020). Dans ce même contexte, on peut également mentionner la stratégie de "error-correcting output-coding" (Dietterich & Bakiri, 1994) et sa généralisation (Allwein *et al.*, 2000). La troisième approche consiste à construire une architecture d'apprentissage hiérarchique en supposant qu'il existe une certaine relation entre les étiquettes de classe. Ainsi, deux stratégies principales ont été proposées, à savoir l'approche basée sur les instances (Zupan *et al.*, 1999), et la stratégie basée sur les prédictions (Godbole *et al.*, 2002; Silva-Palacios *et al.*, 2017).

Dans cet article, nous proposons de tirer le meilleur parti des stratégies par décomposition et hiérarchique en intégrant les relations entre les étiquettes de classe dans une approche de type *one-vs-rest*. À cette fin, nous concevons une **stratégie multitâche**, dans laquelle chaque classifieur *one-vs-rest* est appris dans une configuration multitâche, ce qui permet de prendre en compte l'interdépendance des étiquettes de classe en une seule étape d'apprentissage, par opposition aux stratégies hiérarchiques qui nécessitent de deux étapes interdépendantes. Ainsi, chaque problème (ou tâche) *one-vs-rest* devrait bénéficier de l'apprentissage simultané des autres tâches comme le suggère (Caruana, 1998), si celles-ci sont corrélées ou montrent des similarités cognitives. En particulier, nous proposons deux architectures multitâches de type *one-vs-rest* (**tout-partagé** et **partagé-privé**) basées sur des algorithmes multitâches bien connus (Liu *et al.*, 2017). Ces deux architectures sont comparées à des solutions par décomposition et hiérarchiques de l'état de l'art en utilisant six jeux de données de référence pour l'analyse des sentiments (Socher *et al.*, 2013; Nakov *et al.*, 2016), la détection des émotions (Chatterjee *et al.*, 2019), la classification thématique (Greene & Cunningham, 2006) et la classification de relations lexico-sémantiques (Balikas *et al.*, 2019).

2 Stratégie *one-vs-rest* multitâche

La stratégie par décomposition *one-vs-rest* transforme un problème à N classes en N problèmes binaires (ici, tâches), où chaque classe doit être discriminée de toutes les autres classes. Ainsi, la décision finale est donnée par le classifieur avec la probabilité d'inférence maximale. Dans notre approche *one-vs-rest* multitâche, les N classifieurs binaires sont appris dans un cadre multitâche, ce qui permet d'obtenir une performance accrue pour chacune des tâches si celles-ci sont liées, i.e. s'il existe une relation (de similarité ou cognitive) entre les classes (Caruana, 1998). Nous proposons d'introduire deux modèles multitâches multiclasses : le modèle tout-partagé et le modèle partagé-privé. Dans l'architecture **tout-partagé** (figure 1), un réseau de neurones apprend une représentation partagée commune à toutes les tâches, à partir de laquelle toutes les décisions des classifieurs binaires doivent être apprises. Dans l'architecture **partagé-privé**, une couche privée est concaténée à la couche partagée, permettant à chaque classifieur binaire d'apprendre sa fonction de décision à partir d'informations spécifiques à la tâche mais aussi communes à toutes les tâches (Liu *et al.*, 2017).

Les deux modèles sont formellement définis comme suit pour exactement une couche générique partagée et une couche privée par classifieur binaire. Soit X_k un vecteur d'entrée [1], la couche partagée

1. $X_1 = X$, où X est le vecteur de texte encodé.

$S(X_k)$ est calculée comme dans l'équation 1, où W_{S^k} est une matrice de poids, b_{S^k} est un vecteur de biais [2], et $k \in [1, K]$ où K est le nombre de couches partagées.

$$S(X_k) = \sigma(W_{S^k} X_k + b_{S^k}) = X_{k+1} \tag{1}$$

La couche privée $H^j(Z_q)$ de chaque classifieur binaire indépendant j, qui résout la tâche T_j ($j \in [1, N]$) est définie dans l'équation 2, où $q \in [1, Q]$ et Q est le nombre de couches cachées. Notez que pour l'architecture tout-partagé, $Z_1 = S(X_K)$, et $Z_1 = S(X_K) \oplus X$ pour le modèle partagé-privé, où $\oplus$ représente la concaténation.

$$H^j(Z_q) = \sigma(W^j_{H^q} Z_q + b^j_{H^q}) = Z_{q+1} \tag{2}$$

La classe prédite est donnée par le maximum de toutes les probabilités prédites $O^1, ..., O^N$, qui sont définies dans l'équation 3.

$$O^j = \sigma(W^j_O H^j(Z_Q) + b^j_O) \tag{3}$$

Les paramètres sont mis à jour en minimisant l'entropie croisée binaire $E(O^j, y^j)$, où y^j est l'étiquette de référence. Ainsi, les poids de la couche partagée sont mis à jour en minimisant alternativement l'entropie croisée binaire de chaque classifieur, tandis que les couches privées ne sont mises à jour que pour une tâche donnée.

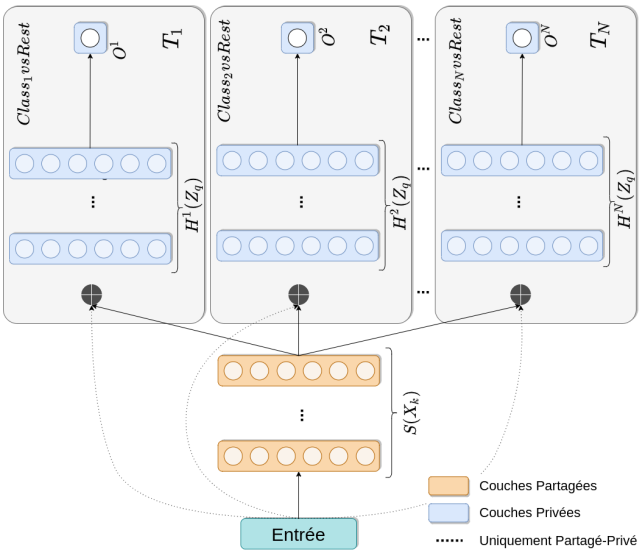

FIGURE 1 – Architectures tout-partagé et privé-partagé de la stratégie *one-vs-rest* multitâche. Les connexions en pointillés ne sont présentes que dans l'architecture partagé-privé.

3 Jeux de données et configurations expérimentales

Notre évaluation porte sur six jeux de données qui s'articulent autour de quatre applications connues de classification textuelle : l'analyse de sentiments, la détection des émotions, la classification thématique, l'identification de relations lexico-sémantiques. Pour l'analyse des sentiments, nous utilisons le jeu de données *Rotten Tomatoes* [3] (Socher *et al.*, 2013), qui consiste en des critiques de films notées sur une échelle de cinq valeurs, et le jeu de données Tweet2016 [4] (Nakov *et al.*, 2016), qui

2. Nous gardons la même notation pour le reste de l'article.
3. https ://www.kaggle.com/c/sentiment-analysis-on-movie-reviews
4. http ://alt.qcri.org/semeval2016/task4/

consiste en des tweets portant un avis sur les principales compagnies aériennes américaines (notation sur une échelle de cinq valeurs). Pour la détection des émotions, nous utilisons le jeu de données EmoContext [5] (Chatterjee *et al.*, 2019), qui consiste en des dialogues agent/utilisateur classés suivant quatre émotions (*happy*, *sad*, *angry*, *others*). Pour la classification thématique, nous utilisons les jeux de données BBC et BBC Sport [6] (Greene & Cunningham, 2006), qui consistent en des articles de presse classés suivant cinq domaines d'actualité, soit généraux pour BBC soit sportifs pour BBC Sport. Pour l'identification des relations lexico-sémantiques, nous créons un jeu de données spécifique RRW en concaténant trois jeux existants : RUMEN [7] (Balikas *et al.*, 2019), ROOT9 [8] (Santus *et al.*, 2016) and WEEDS [9] (Weeds *et al.*, 2014). RRW conjugue ainsi les relations d'hyperonymie, de co-hyponymie, de méronymie et de synonymie. Le détail est présenté dans le Tableau 1.

Jeu de données	Distribution des classes				
Rotten Tomatoes	*negative* 313	*somewhat negative* 686	*neutral* 508	*somewhat positive* 718	*positive* 402
Tweet2016	*very negative* 138	*negative* 2201	*neutral* 10081	*positive* 7830	*very positive* 382
BBC	*business* 510	*entertainment* 386	*politics* 417	*sport* 511	*tech* 401
BBC Sport	*athletics* 101	*cricket* 124	*football* 265	*rugby* 147	*tennis* 100
EmoContext	*happy* 4243	*sad* 5463	*angry* 5506	*others* 14948	- -
RRW	*co-hyponymy* 5283	*hypernymy* 12004	*meronymy* 2943	*random* 25741	*synonymy* 6728

TABLE 1 – Distribution des classes par jeu de données.

En ce qui concerne les configurations expérimentales, chaque jeu de données est aléatoirement divisé en trois sous-ensembles, à savoir l'ensemble d'entraînement (50 %), de validation (20 %) et de test (30 %). Afin de prendre en compte le déséquilibre des classes lors de la phase d'apprentissage [10], nous utilisons Adasyn (He *et al.*, 2008) pour équilibrer les instances sur les classes [11]. Au vu du Tableau 1, seul le jeu de données Tweet2016 reçoit ce pré-processus. Afin de coder l'entrée textuelle, nous utilisons l'architecture d'encodage pré-entraînée *Universal Sentence Encoder* [12] (Cer *et al.*, 2018) avec une dimension de 512 pour l'espace de représentation. Pour les relations lexico-sémantiques, chaque paire de mots est encodée par la concaténation des représentations GloVe (Pennington *et al.*, 2014) avec une dimension de 300.

Trois architectures sont implémentées comme bases de référence, une pour chaque paradigme multi-classe : stratégie native (*MultiClass*), par décomposition (*OneVsRest*), et hiérarchique (*Hierarchical*). *MultiClass* est un réseau de neurones de type *feed-forward* avec des couches cachées activées par une fonction sigmoïde, et la couche de sortie par une fonction *softmax*. L'entropie croisée binaire est utilisée comme fonction de perte. *OneVsRest* implémente une série de N réseaux de neurones de type *feed-forward* avec des couches cachées indépendantes. Toutes les couches sont activées par une fonction sigmoïde, l'entropie croisée binaire est utilisée comme fonction de perte, et le processus

5. https ://www.humanizing-ai.com/emocontext.html
6. https ://www.kaggle.com/c/learn-ai-bbc
7. https ://github.com/Houssam93/MultiTask-Learning-NLP/tree/master
8. https ://github.com/esantus/ROOT9
9. https ://github.com/SussexCompSem/learninghypernyms
10. Un problème bien connu des problèmes multiclasses.
11. Les ensembles de validation et de tests restent déséquilibrés.
12. https ://tfhub.dev/google/universal-sentence-encoder/4

de décision se base sur la fonction maximum. *Hierarchical* correspond à l'algorithme proposé par (Silva-Palacios *et al.*, 2017), que nous avons implémenté spécifiquement pour notre recherche.

Nos deux architectures *one-vs-rest* multitâches (OneVsRest Tout-Partagé et OneVsRest Partagé-Privé) reçoivent exactement la même configuration que le *OneVsRest* pour permettre une comparaison équitable. Le nombre de couches cachées ($K \in [1, 2]$ et $Q \in [1, 2]$), d'époques ($[1..100]$) et de neurones ($[5, 20, 50, 100, 150, 200, 300]$) sont des hyperparamètres optimisés par recherche par grille. Les poids sont initialisés avec une distribution uniforme échelonnée (Glorot & Bengio, 2010) et mis à jour à l'aide de Adam (Kingma & Ba, 2014) avec un taux d'apprentissage fixé à 0,001 [13].

4 Résultats

Chaque architecture est évaluée sur 25 exécutions différentes (partitions d'entraînement et de validation différentes mais avec l'ensemble de test identique) et la signification statistique est calculée pour six métriques différentes afin de refléter au mieux les différences entre les architectures. En particulier, nous calculons la F1 Micro, la F1 Macro, la F1 Pondérée, l'ACC Macro (précision), l'AUNU (surface moyenne sous la courbe) et l'AUNP (surface pondérée sous la courbe) (Hand & Till, 2001; Hossin & Sulaiman, 2015). Les résultats sont présentés dans le Tableau 2.

Les résultats montrent clairement que la stratégie*one-vs-rest* multitâche dépasse l'approche classique *one-vs-rest* en tenant compte de la relation entre les classes. Ceci est particulièrement pertinent pour Tweet2016, EmoContext et RRW, où les architectures tout-partagé et partagé-privé dépassent statistiquement les résultats de la stratégie *OneVsRest*. En particulier, pour Tweet2016, des améliorations maximales de 2,7% F1 Micro, 2,4% F1 Macro et 2,9% F1 pondérée peuvent être atteintes. Les améliorations pour RRW sont du même ordre, alors qu'elles sont moins expressives pour EmoContext, mais néanmoins statistiquement pertinentes. Pour *Rotten Tomatoes* et la classification thématique (c'est-à-dire BBC et BBC Sport), la situation diffère légèrement car seule une des architectures multitâches *one-vs-rest* dépasse statistiquement le *OneVsRest* : OneVsRest Tout-Partagé pour *Rotten Tomatoes* et BBC, et OneVsRest Partagé-Privé pour BBC Sport. En particulier, pour BBC Sport, des améliorations maximales de 0,7% F1 Micro, 0,7% F1 Macro et 0,7% F1 pondérée peuvent être atteintes. La différence entre les architectures tout-partagé et partagé-privé peut être due à la force de la relation entre les classes comme expliqué dans (Qureshi *et al.*, 2020). En effet, les architectures tout-partagé semblent être plus performantes que les architectures partagé-privé lorsque les classes sont fortement liées, alors que le contraire se produit lorsque les classes sont moins fortement liées.

La comparaison entre la stratégie multitâche de type *one-vs-rest* et les implémentations *MultiClass* et *Hierarchical* nécessite une analyse plus approfondie. Pour Tweet2016, EmoContext et RRW, la stratégie tout-partagé dépasse statistiquement les deux approches. En particulier, pour Tweet2016, des améliorations maximales de 0,8% (resp. 3%) F1 Micro, 0,8% (resp. 1,9%) F1 Macro et 0,9% (resp. 3,6%) F1 pondérée peuvent être atteintes par rapport à *MultiClass* (resp. *Hierarchical*). Ces résultats sont confirmés pour l'architecture partagé-privé dans le contexte de EmoContext et de RRW, mais pas pour Tweet2016, où les résultats ne peuvent pas être distingués de *MultiClass*. Pour BBC Sport, l'architecture partagé-privé est statistiquement plus performante que les résultats de *MultiClass*, mais les résultats sont similaires à ceux de *Hierarchical*. Il convient de noter que pour ce jeu de données, l'architecture *MultiClass* affiche les pires résultats, ce qui indique clairement la présence d'une relation entre les classes. Pour BBC, les résultats des stratégies multitâches *one-vs-rest* ne sont pas statistiquement supérieurs à ceux des stratégies *MultiClass* et *Hierarchical*, ce qui montre que ce jeu de données est certainement celui qui met en évidence le moins de relations entre les classes. Enfin,

13. Tous les codes sources sont disponibles sur demande.

	Algorithme	F1 Micro	F1 Macro	ACC Macro	AUNU	AUNP	F1 Pondérée
Tweet2016	*MultiClass*	0.514	0.362	0.805	0.663	0.655	0.537
	OneVsRest	0.495	0.346	0.798	0.653	0.641	0.517
	Hierarchical	0.492	0.351	0.797	0.656	0.634	0.510
	OneVsRest Tout-Partagé	**0.522**⋆+	**0.370**⋆†+	**0.809**⋆+	**0.666**⋆†+	**0.661**⋆†+	**0.546**⋆†+
	OneVsRest Partagé-Privé	0.514⋆+	0.362⋆+	0.806⋆+	0.662⋆+	0.653⋆+	0.539⋆+
Rotten Toma.	*MultiClass*	**0.403**	0.339	**0.761**	**0.598**	**0.605**	0.369
	OneVsRest	0.386	0.348	0.754	0.595	0.598	0.367
	Hierarchical	0.388	0.280	0.755	0.580	0.592	0.326
	OneVsRest Tout-Partagé	0.400⋆+	0.338+	0.760⋆+	0.596+	0.603⋆+	0.368+
	OneVsRest Partagé-Privé	0.393	**0.353**†+	0.757	0.600⋆+	0.602⋆+	**0.372+**
BBC	*MultiClass*	0.954	0.951	0.982	0.971	0.971	0.952
	OneVsRest	0.968	0.967	0.987	**0.980**	0.980	0.968
	Hierarchical	**0.969**	0.968	0.987	**0.980**	0.980	**0.969**
	OneVsRest Tout-Partagé	**0.969**⋆	**0.969**⋆	**0.988**⋆	**0.980**⋆	**0.981**⋆	**0.969**⋆
	OneVsRest Partagé-Privé	**0.969**	0.968	**0.988**	**0.980**	**0.981**	**0.969**⋆
BBC Sport	*MultiClass*	0.947	0.941	0.979	0.963	0.965	0.946
	OneVsRest	0.964	0.966	0.986	0.978	0.976	0.964
	Hierarchical	0.970	**0.974**	**0.988**	**0.983**	**0.980**	0.970
	OneVsRest Tout-Partagé	0.951	0.952	0.981	0.970	0.968	0.951
	OneVsRest Partagé-Privé	**0.971**⋆†	0.973⋆†	**0.988**⋆†	**0.983**⋆†	**0.980**⋆†	**0.971** ⋆†
EmoContext	*MultiClass*	0.825	0.808	0.912	0.869	0.866	0.825
	OneVsRest	0.824	0.806	0.912	0.866	0.864	0.823
	Hierarchical	0.821	0.804	0.911	0.868	0.864	0.820
	OneVsRest Tout-Partagé	**0.828**⋆†+	**0.810**⋆†+	**0.914**⋆†+	**0.872**⋆+	**0.869**⋆†+	**0.827**⋆†+
	OneVsRest Partagé-Privé	0.826⋆†+	0.809⋆+	0.913⋆†+	0.870	0.867⋆+	0.826⋆†+
RRW	*MultiClass*	0.587	0.494	0.835	0.686	0.704	0.583
	OneVsRest	0.597	0.51	0.839	0.687	0.702	0.59
	Hierarchical	0.608	0.512	0.843	0.688	0.708	0.59
	OneVsRest Tout-Partagé	0.617⋆†+	0.519†+	**0.847**⋆†+	0.697⋆†+	0.716⋆†+	0.602⋆†+
	OneVsRest Partagé-Privé	**0.618**⋆†+	**0.521**⋆†+	**0.847**⋆†+	**0.698**⋆†+	**0.717**⋆†+	**0.609**⋆†+

TABLE 2 – Moyenne des scores F1 Micro, F1 Macro, ACC Macro, AUNU, AUNP et F1 Pondérée sur 25 exécutions pour tous les jeux de données. La pertinence statistique est basée sur le test t en supposant des variances d'échantillon inégales, et ⋆, † et + mettent en evidence une valeur de $p \leq 0.05$ par rapport aux algorithmes de références *OneVsRest*, *Multiclass* et *Hierarchical* respectivement.

pour *Rotten Tomatoes*, notre modèle partagé-privé démontre des résultats statistiquement supérieurs par rapport à *MultiClass* pour la F1 Macro, où une amélioration de 1,4% peut être atteinte. Mais c'est la seule mesure où cela se produit. En ce qui concerne les autres métriques, nous obtenons des résultats similaires à ceux du *MultiClass* pour les deux versions *one-vs-rest* multitâches. Cependant, des améliorations significatives sont obtenues par les deux architectures multitâches par rapport à *Hierarchical*, à une exception près, avec des améliorations maximales de 1,2% de F1 Micro, 7,3% de F1 Macro et 4,6% de F1 pondérée.

Afin de mieux comprendre le comportements des différentes architectures, nous présentons leurs performances (F1 Micro) à nombre de paramètres identiques pour tous les jeux de données dans la figure 2. Les résultats montrent que l'architecture hiérarchique est celle qui donne de moins bons résultats quelques soit le niveau de paramètres. Inversement, l'architecture *one-vs-rest* partagé-privé met en évidence les résultats les plus élevés et stables indépendemment du nombre de paramètres, en particulier par rapport à la version multiclasse standard (*MultiClass*), surtout pour un niveau de paramètre faible. Ces résultats confirment les hypothèses des réseaux multitâches qui agissent comme des auto-régulateurs et permettent une meilleure généralisation (Caruana, 1998).

5 Conclusions

Dans cet article, nous définissons deux architectures multitâches de type *one-vs-rest* qui combinent des ensembles de classifieurs appris de manière multitâche pour tirer partie de la relation entre classes. En particulier, nous mettons en œuvre deux solutions différentes (tout-partagé et partagé-privé), qui diffèrent légèrement dans leurs hypothèses d'apprentissage. Les résultats obtenus sur six jeux de données de référence s'attaquant à quatre tâches différentes montrent que la stratégie multitâche de type *one-vs-rest* est toujours plus performante que l'algorithme classique de type *one-vs-rest*. L'approche multitâche est également statistiquement plus performante que les réseaux de neurones classiques et les algorithmes hiérarchiques, lorsque (1) les classes sont fortement liées et (2) les jeux de données contiennent un grand nombre d'exemples d'entraînement (voir tableau 1). Dans tous les autres cas, des résultats solides sont obtenus concurrençant clairement les stratégies de référence.

Il est important de noter que ces derniers algorithmes donnent des résultats avec une variance élevée selon le jeu de données en question, alors que notre approche est plus robuste à la variation (effet de bord de l'approche multitâche). Ainsi, nous pensons que notre approche peut se combiner favorablement à des modèles d'apprentissage spécifiquement adaptés à la tâche ou dédiés au texte. En effet, nous proposons une architecture multiclasse générique adaptable à tout problème de classification à plusieurs étiquettes de classe. Des études dans ce sens sont du ressort du travail futur, comme l'implémentation de nouvelles stratégies de décomposition, notamment impliquant la définition de multiples couches partagées pour des tuples de classes. Nous pensons également nous baser sur les résultats récents de (Akhmouch *et al.*, 2021) pour intégrer des caractéristiques dans le processus de décision et en étudier la distribution afin de maximiser une classification multitâche multiclasse.

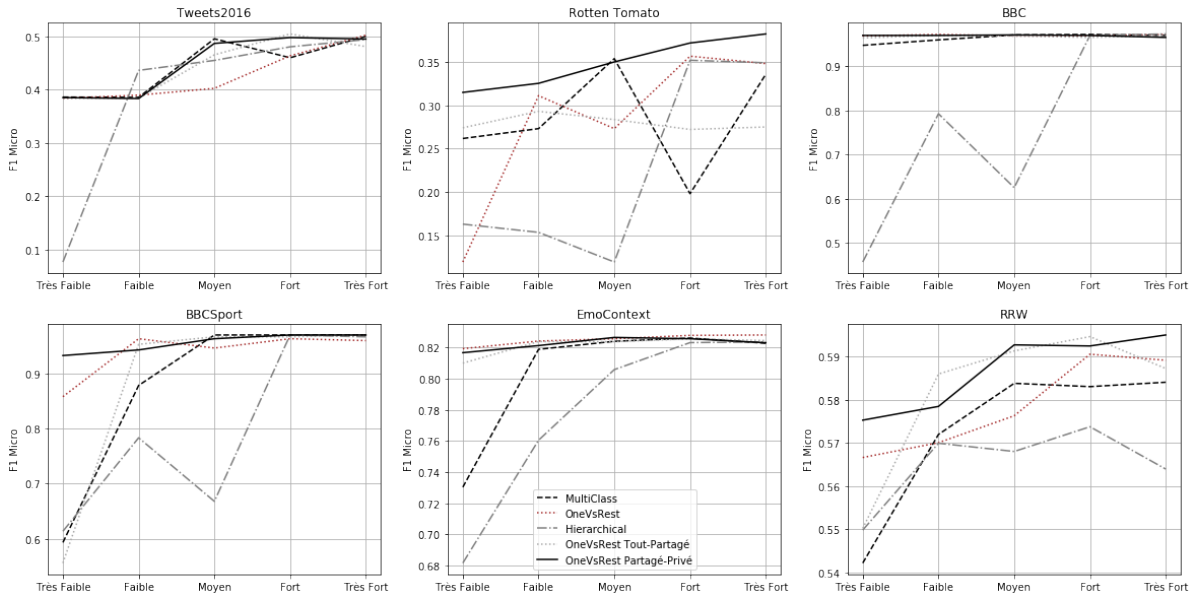

FIGURE 2 – Évolution du F1 Micro pour toutes les architectures avec des paramètres fixes pour tous les jeux de données. Afin de faciliter la lecture et du fait de l'impossibilité de garantir exactement le même nombre de paramètres, 5 niveaux ont été définis pour regrouper la taille des architectures explorées : *Très faible* entre 10k et 30k paramètres, *Faible* entre 30k et 100k, *Moyen* entre 100k et 200k, *Fort* entre 200k et 500k, et finalement, *Très Fort* entre 500k et 1M.

Références

AKHMOUCH H., DIAS G. & MORENO J. G. (2021). Understanding feature focus in multitask settings for lexico-semantic relation identification. In *Proceedings of the 59st Annual Meeting of the Association for Computational Linguistics*.

ALLWEIN E., SCHAPIRE R. & SINGER Y. (2000). Reducing multiclass to binary : A unifying approach for margin classifiers. *Journal of Machine Learning Research*, **1**, 113–141.

BALIKAS G., DIAS G., MORALIYSKI R., AKHMOUCH H. & AMINI M.-R. (2019). Learning lexical-semantic relations using intuitive cognitive links. In *41st European Conference on Information Retrieval (ECIR)*, p. 3–18.

BALIKAS G., MOURA S. & AMINI M.-R. (2017). Multitask learning for fine-grained twitter sentiment analysis. In *40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, p. 1005–1008.

CARUANA R. (1998). Multitask learning. In *Learning to learn*, p. 95–133. Springer.

CER D., YANG Y., KONG S., HUA N., LIMTIACO N., JOHN R. S., CONSTANT N., GUAJARDO-CESPEDES M., YUAN S., TAR C., SUNG Y., STROPE B. & KURZWEIL R. (2018). Universal sentence encoder. *CoRR*, **abs/1803.11175**.

CHATTERJEE A., NARAHARI K. N., JOSHI M. & AGRAWAL P. (2019). SemEval-2019 task 3 : EmoContext contextual emotion detection in text. In *13th International Workshop on Semantic Evaluation (SemEval)*, p. 39–48.

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

CRAMMER K. & SINGER Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, **2**, 265–292.

DIETTERICH T. & BAKIRI G. (1994). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, **2**, 263–286.

GLOROT X. & BENGIO Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, p. 249–256.

GODBOLE S., SARAWAGI S. & CHAKRABARTI S. (2002). Scaling multi-class support vector machines using inter-class confusion. In *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, p. 513–518.

GREENE D. & CUNNINGHAM P. (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering. In *23rd International Conference on Machine Learning (ICML)*, p. 377–384.

HAND D. & TILL R. (2001). A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, **45**(2), 171–186.

HE H., YANG B., EDWARDO G. & SHUTAO L. (2008). Adasyn : Adaptive synthetic sampling approach for imbalanced learning. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, p. 1322–1328.

HOSSIN M. & SULAIMAN M. N. (2015). A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, **5**(2), 1–11.

HSU C.-W. & LIN C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, **13**(2), 415–425.

KINGMA D. & BA J. (2014). Adam : A method for stochastic optimization. *2nd International Conference on Learning Representations (ICLR)*.

LIU P., QIU X. & HUANG X. (2017). Adversarial multi-task learning for text classification. In *55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

NAKOV P., RITTER A., ROSENTHAL S., SEBASTIANI F. & STOYANOV V. (2016). SemEval-2016 task 4 : Sentiment analysis in Twitter. In *10th International Workshop on Semantic Evaluation (SemEval)*, p. 1–18.

OU G. & MURPHEY Y. L. (2007). Multi-class pattern classification using neural networks. *Pattern Recognition*, **40**(1), 4–18.

PAWARA P., OKAFOR E., GROEFSEMA M., HE S., SCHOMAKER L. & WIERING M. (2020). One-vs-one classification for deep neural networks. *Pattern Recognition*, **108**, 107528.

PENNINGTON J., SOCHER R. & MANNING C. D. (2014). Glove : Global vectors for word representation. In *Conference on Empirical Methods on Natural Language Processing (EMNLP)*, p. 1532–1543.

QURESHI S., DIAS G., SAHA S. & HASANUZZAMAN M. (2020). Improving depression level estimation by concurrently learning emotion intensity. *IEEE Computational Intelligence Magazine*, **15**, 47–59.

SANTUS E., LENCI A., CHIU T.-S., LU Q. & HUANG C.-R. (2016). Nine features in a random forest to learn taxonomical semantic relations. In *10th International Conference on Language Resources and Evaluation (LREC)*, p. 4557–4564.

SILVA-PALACIOS D., FERRI C. & RAMÍREZ-QUINTANA M. J. (2017). Improving performance of multiclass classification by inducing class hierarchies. *Procedia Computer Science*, **108**, 1692–1701.

SOCHER R., PERELYGIN A., WU J., CHUANG J., MANNING C., NG A. & POTTS C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 1631–1642.

TENG Z., VO D.-T. & ZHANG Y. (2016). Context-sensitive lexicon features for neural sentiment analysis. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 1629–1638.

WANG G., LI C., WANG W., ZHANG Y., SHEN D., ZHANG X., HENAO R. & CARIN L. (2018). Joint embedding of words and labels for text classification. In *56th Annual Meeting of the Association for Computational Linguistics (ACL)*, p. 2321–2331.

WEEDS J., CLARKE D., REFFIN J., WEIR D. & KELLER B. (2014). Learning to distinguish hypernyms and co-hyponyms. In *25th International Conference on Computational Linguistics (COLING)*, p. 2249–2259.

YAO L., MAO C. & LUO Y. (2019). Graph convolutional networks for text classification. In *33rd Conference on Artificial Intelligence (AAAI)*, p. 7370–7377.

YE Z., GENG Y., CHEN J., CHEN J., XU X., ZHENG S., WANG F., ZHANG J. & CHEN H. (2020). Zero-shot text classification via reinforced self-training. In *58th Annual Meeting of the Association for Computational Linguistics (ACL)*, p. 3014–3024.

ZHANG H. & ZHANG J. (2020). Text graph transformer for document classification. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 8322–8327.

ZHOU S. & LI X. (2020). Feature engineering vs. deep learning for paper section identification : Toward applications in chinese medical literature. *Information Processing & Management*, **57**(3), 102206.

ZUPAN B., BOHANEC M., DEMŠAR J. & BRATKO I. (1999). Learning by discovering concept hierarchies. *Artificial Intelligence*, **109**(1-2), 211–242.