Classification de documents par un réseau de neurones opérant sur des graphes dans l'espace hyperbolique

Adrien Guille, Hugo Attali

Classification de documents par un réseau de neurones opérant sur des graphes dans l'espace hyperbolique

Adrien Guille*, Hugo Attali*

*Université de Lyon, Lyon 2, ERIC UR 3083
adrien.guille@univ-lyon2.fr, hugo.attali@univ-lyon2.fr

Résumé. Diverses architectures de réseaux de neurones sont couramment employées pour la classification de documents, comme les réseaux convolutifs et récurrents, et plus récemment les modèles de langue pré-entraînés basés sur le Transformer. En parallèle, les réseaux de neurones opérant sur les données graphes ont largement progressé. Dans cet article, nous présentons une nouvelle approche où les textes sont encodés individuellement sous la forme de graphes orientés (des sommets mots connectés selon les co-occurrences dans le texte et connectés aux sommets représentant les phrases, eux-même connectés à un sommet document). Nous proposons un réseau de neurones qui apprend à partir de ces graphes et de façon hiérarchique, des représentations des mots, des phrases et du document dans l'espace hyperbolique, dont la courbure permet plus aisément la prise en compte de la hiérarchie que l'espace euclidien. Des expériences poussées montrent l'efficacité de cette approche pour la classification de documents. Notamment, elle s'avère plus performante que le modèle de langue distillé DistilBERT quand elle est entraînée par distillation à partir de BERT-large, bien qu'ayant 160 fois moins de paramètres.

1 Introduction

Diverses architectures neuronales peuvent être employées pour la classification de documents : *e.g.* les réseaux convolutifs (Kim, 2014; Liu et al., 2017), les réseaux récurrents (Yang et al., 2016; Adhikari et al., 2019) ou les réseaux basés sur le Transformer (Vaswani et al., 2017). En particulier, les modèles de langue pré-entraînés basés sur le Transformer, comme BERT (Devlin et al., 2019), peuvent être spécialisés sur un corpus supervisé et obtiennent généralement d'excellent résultats. Toutefois, ayant des centaines de millions de paramètres ces modèles sont coûteux à opérer. Bien que des modèles compressés réduisent le nombre de paramètres à quelques dizaines de millions de paramètres, comme DistilBERT (Victor et al., 2019), leur adoption reste limitée du fait des coûts économiques et environnementaux relativement importants (Bhattacharjee et al., 2020). En parallèle, les réseaux de neurones opérant sur les graphes ont largement progressé (Kipf et Welling, 2017; Veličković et al., 2018; Attali et al., 2023) et les adapter pour la classification de documents est devenu un nouvel axe de recherche (Nikolentzos et al., 2020; Guille et Attali, 2022a,b; Piao et al., 2022).

Proposition Nous encodons les documents individuellement sous la forme de graphes orientés. Chaque graphe est composé d'un sommet document, connecté à des sommets phrases, eux-mêmes connectés à des sommets mots interconnectés d'après les cooccurrences dans le document. Nous proposons une nouvelle architecture neuronale pour classifier les documents d'après ces graphes, qui apprend de manière hiérarchique des représentations des mots, des phrases et des documents. Les représentations sont apprises dans l'espace hyperbolique, dont la courbure permet plus aisément de capturer la structure hiérarchique des graphes traités par le réseau. Ce réseau se distingue par ailleurs des autres réseaux opérant sur des graphes pour la classification de documents, dans la mesure où il ne nécessite pas de fonction d'agrégation globale grâce à l'incorporation d'un sommet document, dont la représentation est directement utilisée pour la classification. Nous nommons cette approche HH-GNN, l'acronyme pour *Hyperbolic Hierarchical Graph Neural Network* [1].

Résultats Nous évaluons cette approche sur 5 jeux de données, de tailles variées et composés de documents de natures différentes (articles journalistiques, brèves, articles scientifiques, posts de forum). Elle se révèle plus efficace que des réseaux représentatifs, de type convolutifs, récurrents et adaptés aux graphes. Nous diagnostiquons également l'effet des différents composants de l'architecture HH-GNN et mettons en avant leur importance. En particulier nos expériences valident l'intérêt de modéliser les mots, les phrases et documents dans l'espace hyperbolique plutôt que l'espace euclidien. Bien que le modèle de langue BERT-large obtienne de meilleurs résultats, nous montrons que notre approche peut dépasser DistilBERT en distillant (Hinton et al., 2015) la connaissance de BERT-large tout en ayant environ 160 fois moins de paramètres.

2 État de l'art

Réseaux convolutifs et récurrents Kim (2014) propose de calculer des convolutions 1D sur des séquences de représentations de mots, suivies d'un sous-échantillonage par maximum global. XML-CNN rend ce réseau plus expressif en appliquant un sous-échantillonage par maximum local, ce qui nécessite une couche additionnelle de réduction de dimension (Liu et al., 2017). Bien qu'efficaces, ces réseaux sont limités car ils ne peuvent capturer que des relations à courte distance dans le texte. Les réseaux récurrent basés sur les cellules LSTM/GRU tentent de capturer des relations à plus longue distance en propageant des états cachés le long des séquences entières de représentations. Yang et al. (2016) utilisent la cellule GRU pour mettre au point un réseau hiérarchique pour la classification de documents. Adhikari et al. (2019) proposent avec le Reg-LSTM d'améliorer la performance de la cellule LSTM pour la classification de documents, en incorporant des mécanismes de "weight dropping" et "embedding dropout".

Transformers L'architecture proposée par Vaswani et al. (2017), un réseau de neurones profond à propagation avant basé sur le mécanisme d'attention a permis la mise au point de modèles de langue pré-entraînés tels que BERT (Devlin et al., 2019), comportant des centaines de millions de paramètres, puis de modèles compressés comportant des dizaines de millions de

1. Code pour HH-GNN : `https://github.com/AdrienGuille/HH-GNN`

paramètres, comme DistilBERT (Victor et al., 2019). Bien que BERT obtiennent généralement de très bon résultats en classification après spécialisation, ses coûts tant opérationnels qu'environnementaux sont prohibitifs (Strubell et al., 2019). Les modèles compressés quant à eux n'obtiennent pas nécessairement des résultats qui justifient leurs coûts, même moindres.

Réseaux de neurones sur graphes En parallèle les réseaux de neurones opérant sur des données structurées en graphe ont largement progressé, les approches les plus connues étant GCN (Kipf et Welling, 2017) et GAT (Veličković et al., 2018). Yao et al. (2019) sont parmis les premiers à adapter le GCN pour la classification avec TextGCN. Leur approche est transductive et nécessite que tous les documents, y compris les documents non étiquetés, soient connus à l'entraînement. Elle consiste à décrire la composition des documents et les cooccurrences à l'échelle entre mots à l'échelle du corpus dans un seul grand graphe et à résoudre la tâche de classification de manière semi-supervisée. TextING (Zhang et al., 2020) et MPAD (Nikolentzos et al., 2020) sont des approches quant à elles inductives où chaque document est encodé individuellement sous la forme d'un graphe décrivant les cooccurrences entre mots à l'intérieur des documents. Ces deux méthodes traitent les graphes de façon similaire, les représentations des mots étant propagées à travers plusieurs sous-réseau impliquant des cellules GRU et des perceptron multi-couches. La propagation terminée, les représentations de tous les sommets du graphe sont agrégées par une fonction aditionnelle pour obtenir la représentation du document à classifier. Plus récemment, Piao et al. (2022) ont proposé TextSSL, qui opère sur des graphes qui incluent en plus des sommets pour modéliser les phrases et ainsi avoir un encodage plus fins des documents. Le réseau apprend ensuite à enrichir ces graphes par l'ajout des arêtes entre mots de phrases différentes. Bien que cela lui confère potentiellement une bonne capacité d'adaptation à des documents de natures diverses, cela à un coût car il est nécessaire de rechercher les valeurs de plusieurs hyper-paramètres, notamment une valeur de température et un seuil pour contrôler l'ajout d'arêtes.

Géométrie hyperbolique pour les réseaux de neurones Représenter les sommets de graphes sans échelle ou hiérarchiques dans l'espace euclidien engendre une distorsion (Chen et al., 2013). En comparaison, l'espace hyperbolique – de par sa courbure – est plus adapté. Par conséquent, des adaptions hyperboliques de réseaux pour graphes ont été proposées. Ceci n'est pas trivial dans la mesure où les opérations basiques comme l'addition vectorielle ou le produit vecteur-matrice ne se généralisent pas simplement à cet espace (Ganea et al., 2018). Chami et al. (2019) proposent un réseau apprenant des représentations sur une surface hyperboloïde dont la courbure constante négative est un paramètre ; les opérations sur ces représentations sont réalisées localement via leur projection dans l'espace tangent.

Nos travaux se basent sur ces avancées récentes, avec pour objectif de mettre au point un réseau de neurones compact opérant sur des documents encodés comme des graphes.

3 Modèle proposé

3.1 Encodage des documents

Soit un document comportant n_p phrases et n_m mots. Nous l'encodons sous la forme d'un graphe orienté avec $N = 1 + n_p + n_m$ sommets, à savoir un sommet pour le document, un

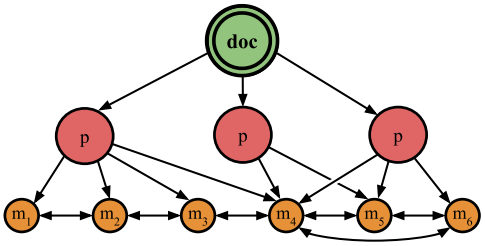

FIG. 1 – *Encodage d'un document (p : phrase, m_i : mot).*

sommet par phrase et un sommet pour chaque mot distinct. Le sommet document (numéroté 0) est connecté par des arcs sortants aux phrases (numérotées 1 à n_p), tandis que les phrases sont connectées aux mots qui les composent par des arcs sortants. Chaque mots est connectés au mot qui le précède et au mot qui le suit directement dans la même phrase. Ce graphe est caractérisé par sa matrice d'adjacence $\mathbf{A} \in \{0;1\}^{N\times N}$. Les attributs des sommets, des identifiants les associant à leur représentation initiale dans le réseau de neurones, sont listés dans la matrice $\mathbf{F} \in \mathbb{N}^{N\times 1}$. Le sommet document est associé à un identifiant spécial, 0, et les phrases sont toutes associées à un autre identifiant spécial, 1. La Fig. 1 montre un exemple de graphe.

3.2 Architecture du modèle

Le réseau reçoit en entrée la matrice d'adjacence et les identifiants associés aux sommets. Les identifiants sont passés à une couche de représentation euclidienne, typiquement initialisée avec des représentations pré-entraînées, suivie d'une couche de projection dans l'espace hyperbolique. Ces représentations ainsi que la matrice d'adjacence sont ensuite passées à L couches d'attention successives, avec des connexions résiduelles sur les poids d'attention. Enfin, la seule représentation du sommet document est extraite et passée à une couche dense pour réaliser la classification. La Fig. 2 illustre l'architecture générale du réseau. Nous détaillons chaque couche ci-après.

3.2.1 Couche de représentation euclidienne

Cette couche est paramétrée par $\mathbf{X} \in \mathbb{R}^{|V|\times d}$ et associe chaque identifiant à un vecteur $\mathbf{x} \in \mathbb{R}^d$ (avec V le vocabulaire et d un hyperparamètre).

3.2.2 Couche de représentation hyperbolique

Cette couche est paramétrée par un scalaire $K > 0$. Elle associe chaque représentation euclidienne $\mathbf{x}_i \in \mathbb{R}^d$ reçue en entrée à un point $\mathbf{y}_i$ dans $\mathbb{H}^{d,K}$, la variété hyperboloïde avec une courbure négative constante de $-1/K$. On note $\mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$, l'approximation de premier ordre autour d'un point quelconque $\mathbf{x}$ de la variété, autrement dit l'espace tangent en $\mathbf{x}$. Considérant $[0, \mathbf{x}_i]$ comme un point dans l'espace tangent $\mathcal{T}_{\mathbf{o}}\mathbb{H}^{d,K}$, on projette $\mathbf{x}_i$ dans $\mathbb{H}^{d,K}$ via la carte exponentielle en utilisant $\mathbf{o} = \{\sqrt{K}, 0, 0, ..., 0\} \in \mathbb{H}^{d,K}$ (l'origine de la variété) comme point

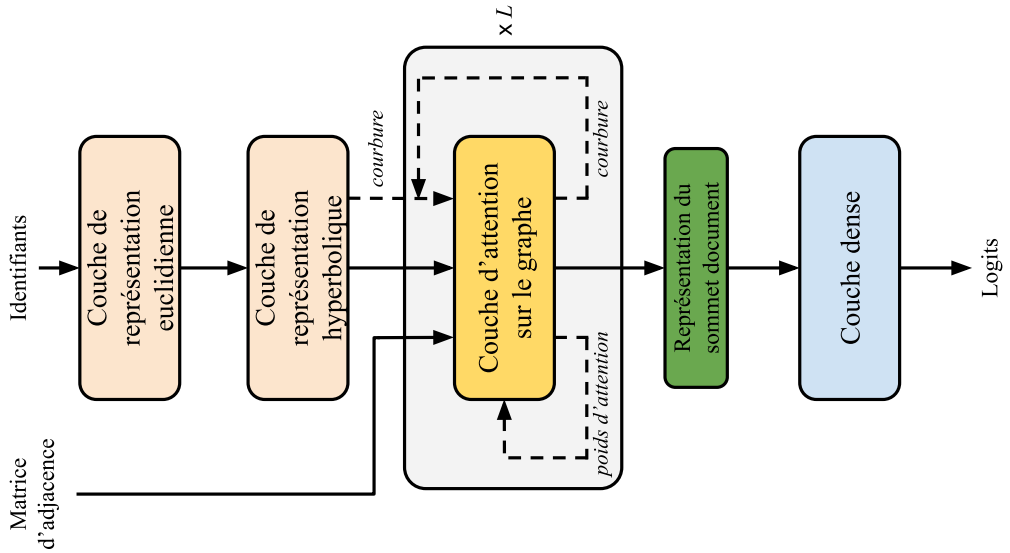

FIG. 2 – *Architecture du modèle (entrée du réseau à gauche, sortie à droite).*

de référence :

$$\mathbf{y}_i = \exp_{\mathbf{o}}^K(\mathbf{x}_i) = \left[\sqrt{K}\cosh\left(\frac{\|\mathbf{x}_i\|}{\sqrt{K}}\right), \sqrt{K}\sinh\left(\frac{\|\mathbf{x}_i\|}{\sqrt{K}}\right)\frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}\right]. \tag{1}$$

L'article de Chami et al. (2019) donne plus de détails concernant la carte exponentielle.

3.2.3 Couche d'attention sur le graphe avec connexion résiduelle

Entrée La couche ℓ ($1 \geq \ell \geq L$) reçoit les représentations hyperboliques calculées à la couche précédentes $\{\mathbf{y}_i^{\ell-1} \in \mathbb{H}^{d,K^{\ell-1}}\}$. $K^{\ell-1}$ fait référence au paramètre de courbure de la couche précédente, avec K^0 et $\mathbf{y}_i^0$ la courbure et les représentations calculées dans la couche de représentation hyperbolique.

Transformation linéaire Pour transformer linéairement les représentations hyperboliques, nous les projetons sur le plan tangent à l'origine de la variété puis les multiplions par la matrice $\mathbf{W}^\ell \in \mathbb{R}^{d\times d}$. Pour une représentation donnée $\mathbf{y}_i^{\ell-1} \in \mathbb{H}^{d,K^{\ell-1}}$, nous calculons $\mathbf{x}_i^\ell \in \mathcal{T}_{\mathbf{o}}\mathbb{H}^{d,K^{\ell-1}}$ ainsi :

$$\mathbf{x}_i^\ell = \mathbf{W}^\ell \log_{\mathbf{o}}^{K^{\ell-1}}\left(\mathbf{y}_i^{\ell-1}\right). \tag{2}$$

La projection est faite via la carte logarithmique, définie ainsi pour un vecteur $\mathbf{y} \in \mathbb{H}^{K,d}$ (cf. Chami et al. (2019)) :

$$\log_{\mathbf{o}}^K(\mathbf{y}) = \sqrt{K}\cosh^{-1}\left(\frac{\mathbf{y}_{[1]}}{\sqrt{K}}\right) \times \frac{\mathbf{y}_{[2:d+1]}}{\|\mathbf{y}_{[2:d+1]}\|}, \tag{3}$$

où $\mathbf{y}_{[1]}$ désigne le premier coefficient de $\mathbf{y}$ et $\mathbf{y}_{[2:d+1]}$ le vecteur privé du premier coefficient.

Attention sur le graphe avec connexion résiduelle Nous calculons des poids d'attention entre paires de sommets connectés d'après les représentations transformées dans l'espace tangent. Si le sommet i est connecté par un arc au sommet j, nous calculons un poids $\alpha_{ij}^{\ell} \in \mathbb{R}$, paramétré par $\mathbf{w}^{\ell} \in \mathbb{R}^{2d}$:

$$\alpha_{ij}^{\ell} = \frac{\exp\left(\text{ReLU}\left(\mathbf{w}^{\ell} \cdot \left[\mathbf{x}_i^{\ell}, \mathbf{x}_j^{\ell}\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{ReLU}\left(\mathbf{w}^{\ell} \cdot \left[\mathbf{x}_i^{\ell}, \mathbf{x}_k^{\ell}\right]\right)\right)}, \tag{4}$$

où $\mathcal{N}_i$ est l'ensemble des sommets ciblés par des arcs provenant du sommet i, plus le sommet i lui-même. Cette manière de calculer les poids d'attention est semblable à ce que fait la couche GAT (Veličković et al., 2018), toutefois nous remplaçons l'activation LeakyReLU par une simple activation ReLU pour éviter des hyperparamètres additionels.

Dans la première couche d'attention ($\ell = 1$), les représentations des sommets voisins sont agrégées comme suit :

$$\mathbf{h}_i^1 = \sum_{j \in \mathcal{N}_i} \alpha_{ij}^1 \mathbf{x}_j^1. \tag{5}$$

Dans les couches suivantes ($\ell > 1$), nous lissons les poids d'attention en incorporant une connexion résiduelle :

$$\mathbf{h}_i^{\ell} = \sum_{j \in \mathcal{N}_i} \frac{1}{2}\left(\alpha_{ij}^{\ell} + \alpha_{ij}^{\ell-1}\right) \mathbf{x}_j^{\ell}. \tag{6}$$

Cette agrégation diffère de plusieurs manières de l'agrégation décrite par Chami et al. (2019). D'abord, les poids d'attention sont obtenus via une simple fonction linéaire, tandis que Chami et al. emploient un perceptron multi-couches. Ensuite, nous calculons l'agrégation dans le plan tangent à l'origine de la variété, ce qui permet une implémentation parallèle efficace ; Chami et al. calcule l'agrégation relative à chaque sommet dans le plan tangent en la représentation de ce sommet, ce qui est beaucoup plus coûteux. Enfin, nous incluons une connexion résiduelle sur les poids d'attention dans le but de les lisser de couche en couche.

Sortie Enfin, la couche renvoie pour chaque sommet une représentation $\mathbf{y}_i^{\ell} \in \mathbb{H}^{d,K^{\ell}}$ en reprojetant $\mathbf{h}_i^{\ell}$ dans l'espace hyperbolique de courbure $K^{\ell} > 0$, un paramètre de la couche, via la carte exponentielle (voir l'équation 1) :

$$\mathbf{y}_i^{\ell} = \exp_{\mathbf{o}}^{K^{\ell}}\left(\mathbf{h}_i^{\ell}\right). \tag{7}$$

3.2.4 Classifieur

Nous extrayons la représentation du sommet document calculé par la dernière couche d'attention, $\mathbf{y}_0^L$, et la passons à une couche dense avec activation softmax pour la classification multi-classes :

$$\hat{p} = \text{softmax}(\mathbf{y}_0^L \mathbf{C} + \mathbf{b}), \tag{8}$$

où $\mathbf{C} \in \mathbb{R}^{d \times c}$ et $\mathbf{b} \in \mathbb{R}^c$ sont les paramètres du classifieur linéaire, avec c le nombre de classes. Toutes les paramètres du réseau sont estimés avec la variante Adam Kingma et Ba (2014) de la descente de gradient stochastique par mini-batchs, en minimisant l'entropie croisée catégorielle. Dans le cas de la classification multi-labels, nous substituons l'activation softmax par une activation sigmoïde et estimons les paramètres en minimisant l'entropie croisée binaire.

TAB. 1 – *Description des jeux de données.*

	Reuters-90	**Reuters-8**	**Ohsumed**	**20-NG**	**AG-News**
docs (train)	5827	5025	3022	10182	108000
docs (val)	1943	559	335	1132	12000
docs (test)	3019	2208	4043	7532	7600
phrases/doc	6,8	5,8	8,6	13,6	2,3
mots/docs	147,3	123,7	211,1	316,6	44,9
type	articles de presse	articles de presse	articles scientifiques	posts de forum	manchettes de presse
labels	90	8	23	20	4
classification	multilabels	multiclasses	multiclasses	multiclasses	multiclasses

4 Évaluation

4.1 Jeux de données

Nous conduisons des expériences basées sur 5 jeux de données bien connus, de tailles variée et composés et de documents de types différents, décrits dans la Tab. 1.

4.2 Méthodes comparées

HH-GNN Nous évaluons notre approche avec 3 couches d'attention en dimension 300, avec pour courbure initiale $k = 5$. Nous ajoutons des connexions résiduelles sur les représentations entre couches pour limiter le surapprentissage sur les plus petits jeux de données (Reuters-90, Reuters-8 et Ohsumed).

Méthode de références Nous considérons 6 méthodes représentatives, à savoir 2 réseaux convolutifs (Base-CNN, XML-CNN), 2 réseaux récurrents (HAN et Reg-LSTM) et 2 réseaux sur graphes (MPAD et TextSSL) :

— **Base-CNN**[2] (Kim, 2014) : filtres de largeur 3, 4 et 5, 100 filtres par largeur, comme dans l'article original.

— **XML-CNN**[2] (Liu et al., 2017) : filtres de largeur 2, 4 et 8, 100 filtres pour chaque largeur et fenêtre de largeur 8 pour le pooling dynamique comme Adhikari et al. (2019).

— **HAN**[2] (Yang et al., 2016) : 2 réseaux GRU bidirectionnels pour encoder respectivement les phrases et les documents ; nous considérons 2 variantes : une avec des états cachés en dimension 100 (donc 200 en bidirectionnel) et une avec des états cachés en dimension 200 (donc 400 en bidirectionnel).

— **Reg-LSTM**[2] (Adhikari et al., 2019) : réseau LSTM suivants les bonnes pratiques modernes pour la classification de document, régularisé par "embedding dropout" avec un taux de 0,1 et "weight dropping" avec un taux de 0,2 ; nous considérons 2 variantes,

2. Code pour Base-CNN, XML-CNN, HAN et Reg-LSTM : `https://github.com/castorini/hedwig`

l'une avec des états cachés en dimension 256, l'autre avec des états cachés en dimension 512 comme Adhikari et al. (2019).

— **MPAD** [3] (Nikolentzos et al., 2020) : 2 couches de propagation de messages, chacune suivie par un MLP à 2 couches qui réduit la dimension à 64 comme Nikolentzos et al. (2020) ; nous considérons également une variante en dimension 128.

— **TextSSL** [4] (Piao et al., 2022) : réseau 2 couches opérant sur le réseaux des cooccurrences dans une fenêtre de taille 3 comme dans le papier original ; nous considérons 2 variantes, l'une qui apprend des représentations de dimension 256, l'autre des représentations de dimension 512. La température et le seuil optimaux sont recherchés parmis $\{0,01; 0,1; 0,5\}$ et $\in \{0,5; 0,75\}$ respectivement.

Nous initialisons HH-GNN et tous ces modèles avec des représentations pré-entraînées sur le corpus Common Crawl via la méthode GloVe en dimension 300 [5]. Tous les réseaux sont entraînés de bout-en-bout, avec un dropout de taux 0,5 avant la couche de classification, en utilisant la variante Adam de la descente de gradient stochastique par mini batchs avec un pas d'apprentissage initial de 0,001. La seule exception est TextSSL, méthode pour laquelle nous recherchons pour chaque jeu de données le meilleur pas d'apprentissage parmi $\{0,0001; 0,0005; 0,001\}$, son apprentissage étant très sensible à cette valeur. Nous considérons aussi 3 méthodes ne nécessitant pas d'être initialisées par représentations pré-entraînées des mots :

— **Régression logistique** [6] : pondération tf-idf et pénalité L2.

— **FastText** [7] (Joulin et al., 2017) : classifieur linéaire sur des moyennes de représentations d'unigrammes et bigrammes de mots en dimension 20.

— **HyperText** [8] (Zhu et al., 2020) : même configuration que FastText, sauf que les représentations sont calculées dans l'espace hyperbolique.

4.3 Résultats principaux

La Tab. 2 liste les scores moyens en 10 répétitions. Notre approche obtient les meilleurs scores sur 3 des 5 jeux de données et le deuxième meilleur score sur les 2 autres jeux de données. La Tab. 3 liste le rang moyen de chaque méthode d'après ces scores. Il apparaît que notre approche est la plus constante avec un rang moyen de 1,4, contre un rang moyen de 4 pour les deux méthodes suivantes, Base-CNN et TextSSL. Ceci montre que l'architecture de base de HH-GNN convient à la résolution de tâches de classification sur des corpus divers.

4.4 Nombre de paramètres

En plus du rang moyen, la Tab. 3 liste le nombre de paramètres de chacune des méthodes – sans compter la couche de représentation euclidienne (commune à tous les modèles) ni la couche de classification (propre à chaque corpus). HH-GNN est le deuxième modèle le plus compact. La seule méthode ayant moins de paramètres est MPAD (dimension 64), toutefois

3. Code pour MPAD : `https://github.com/giannisnik/mpad`
4. Code pour TextSSL : `https://github.com/qkrdmsghk/TextSSL`
5. Représentations GloVe : `https://nlp.stanford.edu/data/glove.42B.300d.zip`
6. Code pour la régression logistique : `https://www.csie.ntu.edu.tw/~cjlin/liblinear/`
7. Code pour FastText : `https://fasttext.cc`
8. Code pour HyperText : `https://github.com/huawei-noah/Pretrained-Language-Model/`

TAB. 2 – *Scores moyens en test (écart-type entre parenthèses). Les meilleurs scores sont en gras et les deuxième meilleurs scores sont soulignés.*

	Reuters-90	**Reuters-8**	**Ohsumed**	**20-NG**	**AG-News**
LR	84,0 (0,0)	96,8 (0,0)	65,5 (0,0)	84,1 (0,0)	91,9 (0,0)
FastText	80,0 (0,0)	96,6 (0,1)	50,9 (0,2)	73,6 (0,1)	91,6 (0,0)
HyperText	82,8 (0,1)	96,4 (0,1)	58,1 (0,1)	82,7 (0,1)	92,3 (0,0)
Base-CNN	85,6 (0,3)	97,0 (0,3)	70,3 (0,4)	**86,3** (0,3)	92,7 (0,2)
XML-CNN	85,4 (0,3)	97,2 (0,2)	67,5 (0,6)	85,1 (0,4)	92,5 (0,3)
HAN (100)	78,1 (1,2)	95,5 (0,5)	64,8 (1,9)	83,5 (0,4)	92,7 (0,1)
HAN (200)	79,7 (0,4)	97,0 (0,3)	69,4 (0,9)	*Erreur*	92,7 (0,2)
Reg-LSTM (256)	79,4 (0,4)	96,7 (0,6)	67,6 (0,7)	85,8 (0,4)	92,8 (0,2)
Reg-LSTM (512)	80,6 (0,4)	97,2 (0,3)	69,5 (0,6)	85,6 (0,4)	92,8 (0,1)
MPAD (64)	86,1 (0,3)	96,9 (0,3)	68,4 (1,0)	83,7 (0,5)	92,7 (0,0)
MPAD (128)	86,6 (0,6)	96,9 (0,2)	67,7 (1,3)	84,1 (0,5)	92,7 (0,1)
TextSSL (256)	**89,4** (0,3)	97,3 (0,3)	68,2 (1,3)	85,3 (0,6)	92,7 (0,2)
TextSSL (512)	89,2 (0,3)	97,1 (0,3)	66,8 (2,1)	85,9 (0,4)	92,9 (0,1)
HH-GNN	89,3 (0,2)	**97,5** (0,2)	**70,7** (0,5)	85,9 (0,3)	**93,7** (0,1)

TAB. 3 – *Nombre de paramètres et rang moyen.*

Modèle	**Nombre de paramètres**	**Rang moyen**
XML-CNN	2160k	7,4
Reg-LSTM (512)	1660k	4,4
HAN (200)	1646k	7
TextSSL (512)	1467k	4,4
Reg-LSTM (256)	571k	7
HAN (100)	503k	9,6
MPAD (128)	421k	6,4
TextSSL (256)	406k	4
Base-CNN	360k	4
HH-GNN	272k	**1,4**
MPAD (64)	**116k**	6,4

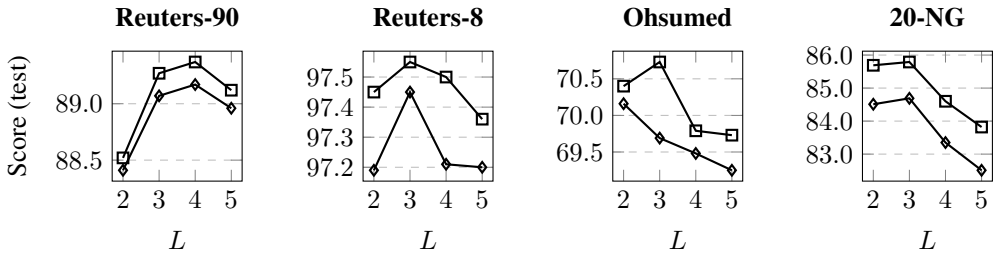

FIG. 3 – *Scores de HH-GNN (—□—) et EH-GNN (—◇—) selon le nombre de couches (L).*

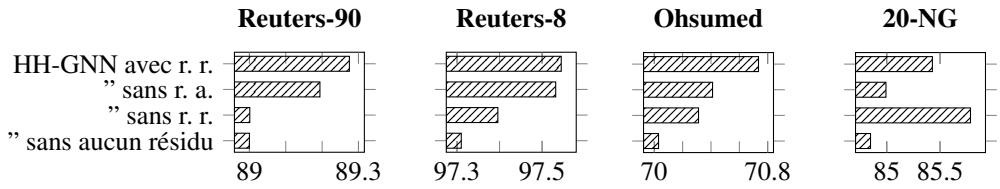

FIG. 4 – *Scores de HH-GNN avec résidus de représentations (” avec r. r.), sans résidus d'attention (” sans r. a.), sans résidus de représentations (” sans r. r.) et sans aucun résidu.*

son rang moyen est nettement moins bon (6,4) et est battue sur tous les jeux de données par HH-GNN (avec une différence de +0,6 à +3,2).

4.5 Variations du modèle

Pour mesurer l'importance des différentes composantes de HH-GNN, nous varions l'architecture de base de différentes manières. La Fig. 3 montre que HH-GNN obtient systématiquement de meilleurs scores que EH-GNN, la variante euclidienne (les représentations et le mécanisme d'attention étant calculés dans l'espace euclidien), ce qui valide empiriquement la pertinence de modéliser les mots, les phrases et les documents dans l'espace hyperbolique. Elle montre également que le meilleur score est atteint, ou presque atteint, avec 3 couches, et qu'augmenter le nombre de couches tend à baisser les scores. La Fig. 4 montre que retirer la connexion résiduelle sur les poids d'attention fait baisser le score. Par ailleurs, elle montre que la connexion résiduelle sur les représentations est bénéfique sur les petits corpus, mais inutile sur les plus grands corpus.

5 Distillation

La Tab. 4 donne les scores obtenus par BERT-large (292 millions de paramètres sans la couche de représentation et la couche de classification) et DistilBERT (42,5 millions de paramètres sans ces 2 couches), ainsi que les scores obtenus par HH-GNN entraîné par distillation (Hinton et al., 2015) de BERT-large, sur les jeux de données augmentés 10 fois selon la technique décrite par Tang et al. (2019). Nous observons que HH-GNN$_{\text{KD}}$ égale ou bat DistilBERT

TAB. 4 – *Modèles de langue et HH-GNN entraîné par distillation (score souligné si supérieur ou égal à DistilBERT).*

	Reuters-90	**Reuters-8**	**Ohsumed**	**20-NG**	**AG-News**
BERT-large	90,4	98,0	75,3	86,5	94,4
DistilBERT	88,8 (0.6)	97,7 (0.2)	69,1 (1.5)	85,6 (0.2)	94,1 (0.1)
HH-GNN$_{\mathbf{KD}}$	<u>89,3</u> (0.2)	<u>97,7</u> (0.2)	<u>72,9</u> (0.3)	<u>86,1</u> (0.2)	93,8 (0.1)

sur 4 des 5 jeux de données bien qu'ayant 160$\times$ moins de paramètres, ce qui en fait une alternative intéressante de par un coût et des temps de latence en inférence nettement inférieurs.

6 Conclusion

Nous avons présenté HH-GNN, un réseau de neurones pour la classification de documents, qui apprend de manière hiérarchique des représentations des mots, des phrases et des documents dans l'espace hyperbolique. L'évaluation a montré l'efficacité de cette approche économe en paramètres. Dans nos futurs travaux, nous aimerions déterminer dans quelle mesure les poids d'attention estimés par le réseaux pourraient permettre d'interpréter ses prédictions.

Références

Adhikari, A., A. Ram, R. Tang, et J. Lin (2019). Rethinking complex neural network architectures for document classification. NAACL.

Attali, H., A. Guille, et S. Chrétien (2023). Amélioration de l'architecture GAT par la prise en compte de la courbure des arêtes du graphe. EGC.

Bhattacharjee, K., M. Ballesteros, R. Anubhai, S. Muresan, J. Ma, F. Ladhak, et Y. Al-Onaizan (2020). To BERT or not to BERT : Comparing task-specific and task-agnostic semi-supervised approaches for sequence tagging. EMNLP.

Chami, I., Z. Ying, C. Ré, et J. Leskovec (2019). Hyperbolic graph convolutional neural networks. NeurIPS.

Chen, W., W. Fang, G. Hu, et M. W. Mahoney (2013). On the hyperbolicity of small-world and treelike random graphs. *Internet Mathematics 9*(4).

Devlin, J., M.-W. Chang, K. Lee, et K. Toutanova (2019). BERT : pre-training of deep bidirectional transformers for language understanding. NAACL.

Ganea, O., G. Becigneul, et T. Hofmann (2018). Hyperbolic neural networks. NeurIPS.

Guille, A. et H. Attali (2022a). Classification interprétable de documents à l'aide d'un réseau de neurones opérant sur des graphes. TextMine @ EGC.

Guille, A. et H. Attali (2022b). Document classification with hierarchical graph neural networks. MLG @ ECML-PKDD.

Hinton, G., O. Vinyals, et J. Dean (2015). Distilling the knowledge in a neural network. DLRL @ NeurIPS.

Joulin, A., E. Grave, P. Bojanowski, et T. Mikolov (2017). Bag of tricks for efficient text classification. EACL.

Kim, Y. (2014). Convolutional neural networks for sentence classification. EMNLP.

Kingma, D. P. et J. Ba (2014). Adam : A method for stochastic optimization. ICLR.

Kipf, T. N. et M. Welling (2017). Semi-Supervised Classification with Graph Convolutional Networks. ICLR.

Liu, J., W.-C. Chang, Y. Wu, et Y. Yang (2017). Deep learning for extreme multi-label text classification. SIGIR.

Nikolentzos, G., A. Tixier, et M. Vazirgiannis (2020). Message passing attention networks for document understanding. AAAI.

Piao, Y., S. Lee, D. Lee, et S. Kim (2022). Sparse structure learning via graph neural networks for inductive document classification. AAAI.

Strubell, E., A. Ganesh, et A. McCallum (2019). Energy and policy considerations for deep learning in NLP. EMNLP.

Tang, R., Y. Lu, L. Liu, L. Mou, O. Vechtomova, et J. Lin (2019). Distilling task-specific knowledge from BERT into simple neural networks. *CoRR abs/1903.12136*.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, et I. Polosukhin (2017). Attention is all you need. NeurIPS.

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò, et Y. Bengio (2018). Graph Attention Networks. ICLR.

Victor, S., L. Debut, J. Chaumond, et T. Wolf (2019). Distilbert, a distilled version of BERT : smaller, faster, cheaper and lighter. EMC2 @ NeurIPS.

Yang, Z., D. Yang, C. Dyer, X. He, A. Smola, et E. Hovy (2016). Hierarchical attention networks for document classification. NAACL.

Yao, L., C. Mao, et Y. Luo (2019). Graph convolutional networks for text classification. AAAI.

Zhang, Y., X. Yu, Z. Cui, S. Wu, Z. Wen, et L. Wang (2020). Every document owns its structure : Inductive text classification via graph neural networks. ACL.

Zhu, Y., D. Zhou, J. Xiao, X. Jiang, X. Chen, et Q. Liu (2020). Hypertext : Endowing fasttext with hyperbolic geometry. EMNLP.

Summary

In this paper, we present HH-GNN, a parameter-efficient graph neural network that operates on documents encoded as tree-like graphs. HH-GNN learns word, sentence and document representations in a hierarchical manner, in the hyperbolic space, whose curvature better fits the structure of these graphs in comparison to the Euclidean space. The evaluation conducted on five well-known datasets against representative CNNs, RNNs and GNNs highlights the relevancy and consistency of HH-GNN. We also show that it can match or outperform DistilBERT when distillating knowledge from BERT-large despite having 160$\times$ fewer parameters.