Architectures Transformeurs pour la classification multilabels de textes

Haytame Fallah, Patrice Bellot, Emmanuel Bruno, Elisabeth Murisasco

Architectures Transformeurs pour la classification multilabels de textes

Haytame Fallah
Aix Marseille Univ, Université de Toulon, CNRS, LIS
Marseille, France
Hyperbios
Toulon, France
haytame.fallah@lis-lab.fr

Patrice Bellot
Aix Marseille Univ, Université de Toulon, CNRS, LIS
Marseille, France
patrice.bellot@univ-amu.fr

Emmanuel Bruno
Université de Toulon, Aix Marseille Univ, CNRS, LIS
Toulon, France
emmanuel.bruno@univ-tln.fr

Elisabeth Murisasco
Université de Toulon, Aix Marseille Univ, CNRS, LIS
Toulon, France
elisabeth.murisasco@univ-tln.fr

ABSTRACT

Les modèles de langue pré-entraînés ont prouvé leur efficacité dans la classification de texte multiclasses. Notre objectif est d'étudier et d'améliorer ce type d'approches pour la classification multilabels de texte, une tâche étonnamment peu explorée au cours de ces toutes dernières années. Cette tâche a pourtant des applications industrielles importantes telles que la recommandation de contenu, l'extraction de méta-données pour l'enrichissement des bases de données ou le routage automatique multicritères des emails. Dans cet article, notre originalité est de proposer des méthodes d'exploitation des activations des couches de sortie des transformeurs pour améliorer la performance de ces modèles pour la classification multilabels. Notre contribution concerne l'évaluation de l'utilité des méthodes de seuillage sur plusieurs modèles d'apprentissage profond, en calculant un seuil de classification global pour optimiser l'ensemble des classes *(SGO)*, ou un seuil individuel propre à chaque classe étudiée *(SI)*. Elle concerne aussi la proposition de deux approches pour la classification multilabels de texte. La première approche (*NPA*) consiste à ajouter un paramètre pour l'apprentissage du nombre de classes et/ou labels N présentes pour un exemple donné, pour considérer les classes qui correspondent aux N activations les plus élevées comme étant des labels valides. La deuxième approche *(TL)* consiste à ajouter une couche au transformeur pour l'apprentissage des critères utiles pour la sélection des labels pertinents. Nous évaluons ces approches sur des corpus d'articles de journaux et d'articles scientifiques. Nous avons aussi constitué et mis à disposition un jeu de données de résumés d'articles scientifiques en français que nous avons conçu à partir du dépôt d'archives ouvertes *'HAL'*. Ces évaluations montrent que la performance de nos propositions dépasse celles des méthodes de l'état de l'art de classification multilabels de texte pour les jeux de données étudiés, et sont transposables à tout problème de classification multilabels utilisant les réseaux de neurones.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Natural language processing**.

KEYWORDS

Apprentissage profond, Classification automatique, Multilabels, Modèles de langue, Transformeurs, BERT.

1 INTRODUCTION

Les modèles de langue probabilistes sont depuis longtemps utilisés pour de très nombreuses tâches du traitement automatique du langage naturel et de la recherche d'information. Nous nous intéressons à l'adaptation de ces modèles pour la classification multilabels où différentes parties du texte contribuent différemment pour la prédiction des labels.

L'adaptation de l'apprentissage profond pour la modélisation séquentielle du langage a débuté avec l'utilisation des réseaux de neurones récurrents (RNN). L'entraînement de ces réseaux se fait d'une manière itérative où un état caché h est mis à jour à partir de chaque nouveau mot d'une séquence d'entrée. Les informations contenues dans chaque composante de la phrase sont ainsi préservées tout au long du traitement de celle-ci, mais avec une contribution qui diminue proportionnellement à la longueur de la séquence.

Les réseaux récurrents de longue mémoire à court terme (LSTM) [Hochreiter and Schmidhuber 1997] ont été conçus pour remédier à ce problème en ajoutant, en plus des états cachés h , des cellules de mémoire qui permettent de réguler le flux d'information et de retenir ou "oublier" les éléments des étapes antérieures selon leur importance. Ce mécanisme a des limitations car l'importance donnée à un mot n'est pas relative au reste de la phrase.

Le mécanisme d'attention [Bahdanau et al. 2016] a été ensuite introduit dans l'utilisation des RNNs. Il permet de donner une importance (poids) relative à chaque mot de la séquence calculée à l'aide d'un réseau de neurones feed-forward (propagation avant). Ces modèles permettent une bonne modélisation du langage et réussissent à capturer la nature séquentielle des phrases, mais avec un temps d'entraînement relativement long. En n'utilisant que le mécanisme d'attention, les transformeurs [Vaswani et al. 2017] peuvent être entraînés en parallélisant le traitement d'une séquence tout en gardant l'information sur l'ordre des mots. Ces transformeurs disposent d'une partie "encodeur" et une autre "décodeur" [Sutskever et al. 2014], composée chacune de plusieurs couches. Les plongements de mots (embeddings), l'entrée de ces modèles, passent par

des couches d'attention (ou self-attention) où ils sont mis à jour en fonction des scores d'attention calculés à partir des autres parties de la séquence, et cela pour avoir une représentation contextualisée pour chaque composante des phrases.

Les transformeurs sont actuellement les meilleures implantations des modèles de langue. Le modèle BERT, pour Bidirectional Encoder Representations from Transformers [Devlin et al. 2019], et ses variantes sont parmi les plus récents. BERT se différencie des autres transformeurs par sa capacité à traiter la séquence de texte d'une manière bi-directionnelle à l'opposé de GPT-2 [Radford et al. 2019] qui ne regarde que la partie gauche du mot en cours de traitement, ou ELMO [Peters et al. 2018] qui concatène la représentation gauche et droite de la séquence, ce qui rend l'apprentissage plus lent. Cette bi-directionnalité de l'apprentissage améliore les représentations linguistiques du modèle car le sens d'un mot de la phrase peut dépendre non seulement des mots qui le précèdent mais aussi de ceux qui le suivent. Ces modèles sont très utiles compte tenu de leur ré-utilisabilité, en pré-entraînant un modèle en non-supervisé sur un large corpus de texte, puis en l'adaptant pour la tâche de traitement de texte souhaitée. Le mécanisme d'attention peut être très performant pour capturer l'importance de certaines parties de l'entrée par rapport à d'autres.

La classification multilabels de texte est une tâche de traitement automatique du langage où un texte en entrée peut être associé à plusieurs classes. Le but étant de pouvoir extraire tous les sujets, prédéfinis ou non, contenus dans ce texte. Plusieurs approches d'apprentissage machine ont été proposées pour aborder ce problème. À notre connaissance, aucun benchmark officiel n'existe pour la classification multilabels de texte, et cela en dépit du fait que cette tâche ait des applications concrètes et importantes dans le monde industriel. Parmi ces applications on retrouve l'indexation multicritères, la recommandation de contenu, ou même l'extraction des méta-données pour l'enrichissement des bases de données applicable sur divers types de corpus, à l'instar des articles scientifiques. Les travaux présentés dans cet article sont réalisés dans le cadre d'une thèse CIFRE en collaboration avec Hyperbios, une société de services informatiques, pour la classification et la segmentation des demandes clients exprimées par mail pour le compte de plusieurs agences d'assurance. La classification multilabels permettra d'identifier les sujets contenus dans chaque mail, et déclencher les actions correspondantes par les systèmes d'automatisation de tâches et de proposition de réponses.

Exemple :

Madame, Suite à notre conversation téléphonique je vous adresse en pièces jointes la photo du compteur kilométrique du véhicule xxxx ainsi que la photo de la plaque d'immatriculation. J'ai bien noté que le véhicule pouvait être conduit par un tiers, et qu'en cas de sinistre il n'y aurait pas de franchise. Je vous remercie de bien vouloir m'adresser aussi rapidement l'attestation d'assurance.

Classifications : Envoi de documents, informations relevé compteur, demande d'attestation.

Cette exemple illustre la complexité du problème qui consiste à extraire tous les labels possibles depuis un texte de taille réduite, ayant peu d'éléments contextuels et sémantiques. Les différents styles d'écritures ainsi que l'emploi de termes qui diffèrent selon le niveau d'expertise du client pour désigner le même concept rajoutent des contraintes supplémentaires à cette problématique.

Dans cet article, nos contributions concernent essentiellement :

- La création d'un corpus de données de classification de texte multilabels en français, contenant les résumés d'articles scientifiques obtenus à partir de l'archive ouverte "*HAL*";
- L'évaluation des performances des modèles de langue disponibles : BERT et ses variantes DistilBERT, RoBERTa et DeBERTa pré-entraînés principalement sur des corpus en anglais, ainsi que CamemBERT et FlauBERT, deux autres variantes de BERT pré-entraînées sur des corpus en français;
- L'étude des méthodes de sélection de seuil pour une exploitation plus efficace des résultats de ces modèles, notamment le choix d'un seuil global qui optimise les performances de toutes les classes, ou le calcul d'un seuil propre à chaque classe pour l'optimisation individuelle des labels;
- La proposition de deux approches alternatives au seuillage pour la sélection des classes pertinentes. La première consiste à introduire un paramètre à la dernière couche du transformeur qui sera entraîné pour le calcul du nombre de classes présentes dans un exemple, la valeur de ce paramètre sera utilisée pour sélectionner les labels ayant les activations les plus fortes; La deuxième consiste à rajouter une couche finale au modèle, qui aura le même nombre de paramètres que l'avant dernière couche (égal au nombre de classes), dans le but d'obtenir des valeurs d'activation plus discriminantes pour l'exemple donné, ie. activation élevée si présence de label, faible dans le cas contraire.

Nous évaluons ces approches sur des corpus comparables d'articles scientifiques, HAL-Dataset pour le français et AAPD pour l'anglais, mais aussi sur des articles d'actualité en anglais (corpus de Reuters). Les modèles et les architectures proposés sont comparés à des approches de référence, performantes et plus transparentes sur les critères de classification appris, comme les arbres de décision et les machines à vecteur support (SVM).

L'article est organisé de la façon suivante : la section 2 présente les approches qui abordent la classification multilabels, les sections 3 et 4 décrivent la méthodologie suivie et les approches proposées et la section 5 est dédiée aux expérimentations.

2 TRAVAUX ANTÉRIEURS

Dans la classification multiclasses, chaque exemple (instance) X du jeu de données est associé à un label unique. La classification multilabels (CMLT) consiste en plus à pouvoir associer chaque entrée avec plusieurs labels $\mathbf{Y}$, plutôt qu'un seul.

2.1 Méthodes de classification multilabels

Les méthodes de classification multilabels peuvent être classées en trois catégories principales : par transformation du problème, par adaptation ou par des méthodes d'ensembles.

2.1.1 Transformation du Problème (PT). La transformation du problème consiste à 'transformer' le jeu de données pour changer le problème en une classification multiclasses à label unique. Une de ces méthodes consiste à considérer toutes les combinaisons uniques de labels possibles du jeu de données, *label powerset* [Tsoumakas et al. 2010], et à entraîner un classifieur multiclasses $M : X \rightarrow P(Y)$, où

$P(Y)$ est le powerset de **Y**, l'ensemble des sous-ensembles uniques et distincts de labels.

En plus du nombre élevé de classes possibles qui peut atteindre $2^{|Y|}$, le challenge réside dans la capacité à trouver suffisamment d'exemples pour chaque combinaison de labels pour éviter la sous-représentation des classes.

La pertinence binaire [Boutell et al. 2004], est une autre méthode de transformation de problème où sont entraînés |**Y**| classifieurs binaires qui détectent la présence ou la non-présence d'un label pour une instance. |**Y**| jeux de données sont construits à partir du jeu original. Chaque jeu de données D_y, $y \in \mathbf{Y}$, contient les instances où y est un label qui les caractérise. Pour une instance x du jeu de données, le résultat de la classification est l'union des labels détectés par chaque classifieur.

Pour un |**Y**| grand, le temps d'entraînement et d'inférence des modèles est élevé. Il est important de noter, qu'en transformant le problème en une classification multiclasses, les dépendances qui peuvent exister entre les différents labels ne sont plus considérées [Luaces et al. 2012].

2.1.2 Méthodes d'ensembles. Un ensemble de classifieurs multiclasses peut être combiné pour créer un classifieur multilabels. Pour une instance donnée, chaque classifieur va prédire une seule classe et toutes les sorties de ces classifieurs sont alors combinées via une méthode d'ensemble. Une de ces méthodes consiste à considérer une classe comme présente si un pourcentage de classifieurs ayant prédit cette classe est atteint, aussi appelé seuil discriminatif [1]. L'algorithme *RAKEL* [Tsoumakas and Vlahavas 2007] est une autre variation de cette méthode. Des classifieurs entraînés sur des sous-ensembles aléatoires des *labels powersets* sont utilisés pour la création d'un classifieur multilabels, les prédictions de ces classifieurs passent par un système de vote pour la prédiction finale.

L'utilisation de plusieurs classifieurs impose des contraintes fortes en termes d'espace mémoire, ainsi que la nécessité d'optimiser un nombre de modèles qui augmente linéairement avec le nombre de classes du jeu de données.

2.1.3 Adaptation du Problème (PA). Les méthodes d'adaptation du problème ne nécessitent pas une transformation du jeu de données mais une adaptation des algorithmes de classification, tels que ML-kNN [Zhang and Zhou 2007] qui étend l'algorithme kNN pour les données multilabels, ou BP-MLL [Min-Ling Zhang and Zhi-Hua Zhou 2006] une adaptation de l'algorithme de rétro-propagation pour les réseaux de neurones.

L'adaptation des algorithmes d'apprentissage profond pour le multilabels reste de manière générale une voie avec peu de contributions. Une adaptation de ces approches pourraient contribuer à une augmentation significative des performances pour la tâche de classification multilabels. L'utilisation d'un seul modèle sans le recours à une transformation préalable des données constitue une méthode efficiente pour essayer de répondre au problème du multilabels.

2.2 Approches de seuillage

Plusieurs méthodes de seuillage ont été proposées pour les approches citées précédemment, ce sont des méthodes qui impactent directement le choix d'une classe pour le problème multilabels. Le seuil peut être ajusté de plusieurs façons, soit pour optimiser toutes les classes (un seuil global), ou pour optimiser individuellement chaque classe (nombre de seuils égal au nombre de classes). Soit m le nombre d'exemples dans le jeu de données de test (ou la validation) et n_y le nombre de classes (labels). Les quatre stratégies les plus utilisées pour le choix du(des) seuil(s) sont :

- **SCut**: Les classes sont optimisées de façon individuelle, les seuils sont choisis en fonction des performances sur le jeu de données de validation, mesurés soit par la maximisation d'un score ou la minimisation d'une fonction de coût, et sans garantir l'obtention d'un optimum global. Cette méthode peut aussi être utilisée pour obtenir un seuil global pour toutes les classes [Yang 2001];
- **RCut** (Rank Cut) : pour chaque instance, les classes sont ordonnées selon le score obtenu, les t premières classes sont choisies comme labels pertinents. Le paramètre t peut être fixé ou réglé à partir du jeu de données de validation [Yang 2001];
- **PCut** (Proportion Cut) : pour chaque classe y_i, les instances du jeu de données de test sont ordonnées selon le score obtenu pour cette classe. Les k_i premières instances sont choisies pour la classe y_i où $k_i = P(y_i) \times x \times n_y$ est le nombre d'instances attribuées à cette classe. $P(y_i)$ est la probabilité qu'une instance fasse partie de la catégorie y_i (calculée préalablement à partir du jeu d'entraînement), et x le nombre moyen d'instance à attribuer pour une classe quelconque fixé au préalable. Si $x = n$ toutes les instances sont prises pour une catégorie, pour $x = 0$ aucun exemple n'est considéré comme faisant partie de la classe étudiée [Lewis et al. 1996; Yang 1997];
- **MCut** (Maximum Cut) : les labels sont ordonnés à partir des scores obtenus pour une instance du jeu de données, le seuil est égal à la moyenne des deux labels contigus pour lesquels l'écart de score est le plus important [**?**].

Des variations de ces méthodes visant à améliorer leur performance et pallier aux problèmes qu'elles peuvent poser ont été suggérées [Yang 2001]. Les méthodes axées sur les scores de classification sont les meilleures parmi les approches de seuillage [**?**]. Nous choisissons dans cet article la méthode SCut pour l'évaluation des modèles de langues étudiés.

2.3 Utilisation de l'apprentissage profond

Les modèles d'apprentissage profond ont aussi été utilisés pour répondre au problème de la classification de texte dont les CNN [Kim 2014], RCNN [Lai et al. 2015] et HAN [Yang et al. 2016] mais sans se focaliser sur le problème multilabels.

[Gan et al. 2019] utilise un réseau de neurones pour la classification multilabels dans le cadre de l'analyse multi-composition spectroscopique, un réseau composé d'une partie classificateur à laquelle un paramètre est ajouté pour l'apprentissage du seuil d'activation optimal pour la classification. Ce paramètre sera optimisé en fonction du seuil calculé en appliquant le modèle en cours d'apprentissage sur le jeu de données d'entraînement, la valeur cible du seuil sera donc différente pour chaque itération de la phase

[1] https://www.scikit-yb.org/en/latest/api/classifier/threshold.html

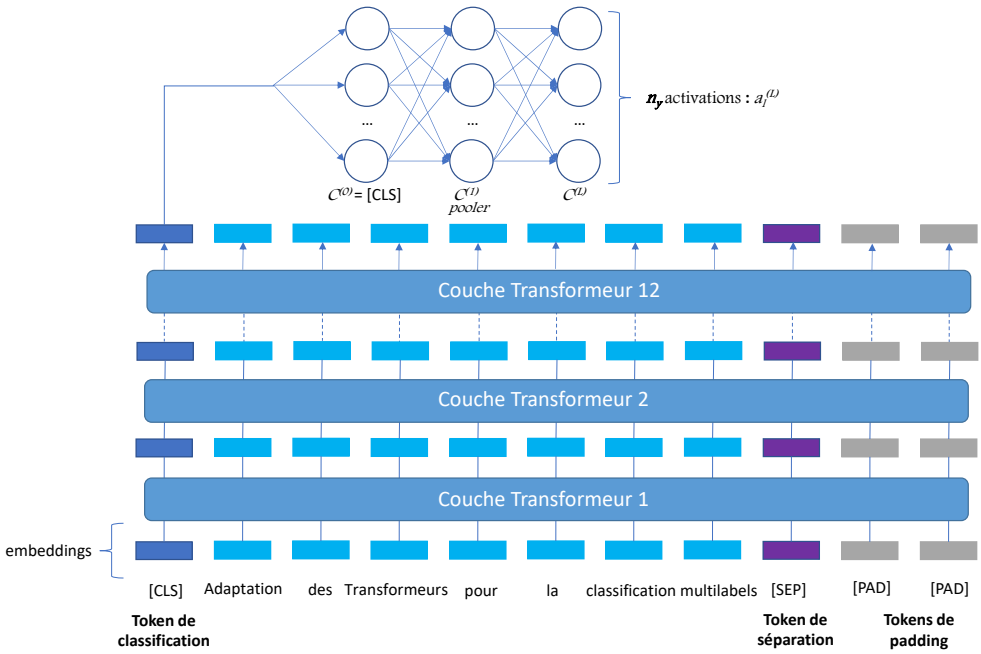

Figure 1: Architecture BERT avec une couche dense de classification au-dessus des couches transformeurs et les valeurs d'activation des sorties.

d'entraînement, ce qui peut engendrer une difficulté de convergence de la fonction d'erreur vers son minimum.

Par ailleurs, MAGNET [Pal et al. 2020] utilise les plongements de BERT [Devlin et al. 2019] comme entrée, il concerne explicitement la CMLT et réussit à avoir de bonnes performances en score F1 [2] pour les jeux de données d'AAPD et de Reuters (cf. section 5.2). DocBERT [Adhikari et al. 2019], qui est maintenant une référence de l'état de l'art, rajoute un réseau linéaire à la tête du transformeur BERT, mais sans traitement par la suite des sorties du modèle.

Les transformeurs ont été utilisés pour la classification multilabels de texte dite 'extrême' [Chang et al. 2019; Gong et al. 2020] où sont traités des corpus très larges de textes ayant un nombre de labels pouvant atteindre les dizaines de milliers. De telles architectures ne sont pas adaptés à la problématique de textes courts.

Les tentatives d'utiliser les réseaux de neurones pour la classification de texte ne se focalisent pas sur la classification multilabels. Celles qui traitent ce problème ne donnent généralement pas d'importance à la manière dont sont exploitées les activations de la couche de sortie (e.g utilisation de seuil) ou en modifiant l'architecture du réseau pour essayer de l'adapter à cette tâche.

3 ADAPTATION DES MODÈLES DE LANGUE PRÉ-ENTRAÎNÉS

Nous nous intéressons dans cet article à l'adaptation des modèles transformeurs, notamment BERT et ses variantes, pour la classification multilabels de texte.

[2] $F1 - score = 2 \times \frac{précision \times rappel}{précision + rappel}$

3.1 Adaptation de BERT pour la classification multilabels

Pour les réseaux de neurones, le texte d'entrée doit être segmenté et converti en une liste de symboles (appelés tokens) pour le traitement. Ces tokens sont par la suite associés à des identifiants uniques, définis dans le dictionnaire utilisé par le modèle, et aux plongements (word embeddings) de mots qui leur correspondent. Ces derniers constituent les données de la couche d'entrée de ces modèles. L'architecture des modèles de langue à base de transformeurs se différencie de leurs prédécesseurs par leur capacité à paralléliser le processus d'entraînement en traitant tout le vecteur (tous les tokens) de l'entrée simultanément. L'entraînement est réalisé en masquant des tokens (de façon aléatoire ou non) et en essayant ensuite de les prédire en utilisant principalement le mécanisme d'attention, processus appelé modélisation masquée du langage.

Le modèle BERT introduit la bi-directionnalité dans la prédiction des tokens masqués, où les deux contextes sémantiques gauche et droit du mot à prédire sont pris en compte. En plus de la modélisation masquée du langage, BERT est entraîné pour la prédiction de la phrase suivante, une tâche où le modèle reçoit une paire de phrases et essaie de prédire si la deuxième phrase de la paire suit la première dans le texte original.

BERT introduit aussi un token spécial de classification **[CLS]** (possédant aussi un identifiant et un vecteur d'embedding) contenant un *état caché* de la phrase, mis à jour dans chacune des couches du modèle. Un réseau linéaire de L couches denses (généralement L=2) constitue les dernières couches du modèle avec le token **[CLS]** comme entrée, et n_y sorties (approche similaire à [Adhikari et al. 2019]. La figure 1 présente l'architecture du modèle.

Le choix des fonctions d'activation et des fonctions de coût dépend de la nature de la tâche de classification. Dans le cas de la classification multiclasses où les classes sont mutuellement exclusives (une instance doit appartenir à une et une seule classe) on utilise classiquement la fonction *Softmax* comme fonction d'activation de la dernière couche du modèle ($C^{[L]}$). Elle a pour but de convertir le vecteur de sorties, les activations de $C^{[L]}$, en un vecteur de probabilités proportionnelles aux valeurs de ces activations. L'entropie croisée (Cross Entropy) est utilisée comme fonction d'erreur du modèle.

Dans le cas de la classification multilabels, la nature de l'architecture des réseaux de neurones peut être exploitée de telle manière à utiliser les valeurs des activations $A^{[L]}$ de la couche de sorties $C^{[L]}$ pour déterminer la présence ou non d'un label, et cela en introduisant un seuil. Ce seuil, s'il est dépassé, permettra de considérer ou non le label étudié. Le but n'étant pas d'avoir une distribution de probabilités en fonction de toutes les classes, la fonction *Softmax* n'est pas adaptée. La fonction d'activation *Sigmoid* σ est plus appropriée pour cette tâche. Elle convertit chaque activation de $A^{[L]}$ en une valeur comprise entre 0 et 1, calcule une probabilité de présence de chaque label y_i en fonction des valeurs de l'activation qui lui correspond $a_{y_i}^{[L]}$ de la couche L. Cela implique l'utilisation de l'entropie croisée binaire (Binary Cross Entropy - BCE) comme fonction de coût qui vise à réduire l'écart entre $a_{y_i}^{[L]}$ et la vraie valeur de sortie (0 ou 1 selon la présence ou non du label y).

Cette approche ne peut être considérée complètement comme une transformation du problème car la méthode ne requiert pas une transformation du jeu de données ou la création de multiples classifieurs binaires, ni une adaptation complète de l'apprentissage profond pour la classification multilabels car les résultats du réseau de neurones doivent être traités en aval pour avoir les classifications finales.

3.2 Apprentissage profond et méthodes de seuillage

Les approches de seuillage peuvent être appliquées au transformeur, si l'on considère que chaque activation de la couche finale est un classifieur binaire de la classe qu'il représente.

3.2.1 Seuil global de classification (SGO). Le seuil de classification s peut être choisi de façon à maximiser l'ensemble des scores de classification. Pendant l'entraînement du modèle et après chaque itération, la valeur de s est variée de 0 à 1, avec un pas défini au préalable (10^{-2}). Le score micro-F1 est ensuite calculé pour chaque seuil s. On obtient au final le seuil global optimal s_{go} qui permet d'obtenir les meilleures performances sur le jeu de données d'entraînement. Ce seuil est ensuite utilisé pour les jeux de données de validation et de test. L'ensemble des labels L présents pour un exemple x peut être exprimé sous la forme:

$$Y_{x \in X} = \cup_{y \in Y} \{y_i\} : \sigma(a_{y_i}^{[L])}) \geq s_{go}$$

$a_{y_i}^{[L]}$ étant l'activation qui correspond au label y_i parmi les activations $A^{[L]}$.

Une autre variation la méthode SCut (avec seuil global) consiste à affiner le seuil optimal à partir du jeu de données de la validation pour ensuite l'appliquer au jeu de données de test. Seule la première approche a été étudiée dans cet article.

3.2.2 Seuils individuels (SI). L'utilisation d'un seuil s commun pour toutes les classes suppose que les descripteurs et les poids directement liés aux activations $A^{[L]}$ soient les mêmes pour chaque classe/label. Ce qui n'est pas le cas, l'intensité d'activation d'un neurone de la dernière couche $a_{y_i}^{[L]}$, lors de la présence du label y pour une instance donnée, varie d'une classe à l'autre, un effet qui s'accentue si le jeu de données est peu équilibré.

Nous proposons donc d'évaluer la méthode SCut (avec seuils individuels) en mettant en place un seuil s_y pour chaque label y du jeu de données, attribué aux activations $a_{y_i}^{[L]}$ de $C^{[L]}$. Les valeurs de s_y sont les valeurs qui maximisent les scores de classification pour un label y. Ces seuils sont calculés comme pour le s_{go} pendant la phase d'entraînement du modèle, et sur le jeu de données d'entraînement, en faisant varier chaque seuil de façon à maximiser le score F1 de chaque label.

$$Y_{x \in X} = \cup_{y \in Y} \{y_i\} : \sigma(a_{y_i}^{[L]}) \geq s_y$$

Comme pour le seuil global, les seuils individuels peuvent être calculés à partir du jeu de données de validation.

4 APPROCHES PROPOSÉES

Nous proposons deux méthodes alternatives au seuillage pour une exploitation plus efficace des valeurs des activations $A^{[L]}$ de la couche de sortie. Ces approches ont pour but de s'affranchir d'un calcul de seuil en apprenant des caractéristiques propres au texte permettant la bonne sélection des labels pertinents.

4.1 N plus grandes activations (NPA)

L'utilisation d'un seuil pour déterminer la présence ou non d'un label entraîne dans plusieurs cas une sous classification (respectivement sur classification) d'une instance quand le nombre prédit de labels est inférieur (respectivement supérieur) au nombre effectif de labels présents. La première alternative proposée consiste à introduire un paramètre (neurone) à la dernière couche de classification et qui sera utilisé uniquement pour le calcul du nombre de labels y présents pour une instance. Soit $A'^{[L]}$ la liste des N plus grandes activations pour un exemple donné, N étant le nombre effectif des labels présents pour cet exemple :

$$Y_{x \in X} = \cup_{y \in Y} \{y_i\} : a_{y_i}^{[L])} \in A'^{[L])}$$

L'optimisation de ce paramètre se fera en utilisant le nombre de classes présentes pour une instance comme valeur cible, et une fonction d'erreur adaptée au problème de régression (par ex. l'erreur absolue moyenne **MAE** ou l'erreur quadratique moyenne **MSE**).

Nous utilisons un optimiseur unique pour la propagation arrière (calculs des gradients des fonctions d'erreurs et mise à jour des poids du modèle), par conséquent, l'erreur de la régression doit être mise à la même échelle que l'erreur de la classification, ceci est fait en réduisant l'erreur de régression par un facteur de 5. La valeur de ce paramètre sera utilisée pour récupérer les N plus grandes activations $a_{y_i}^{[L]}$ qui seront considérées comme labels prédits. La figure 2 présente l'architecture du modèle pour cette approche.

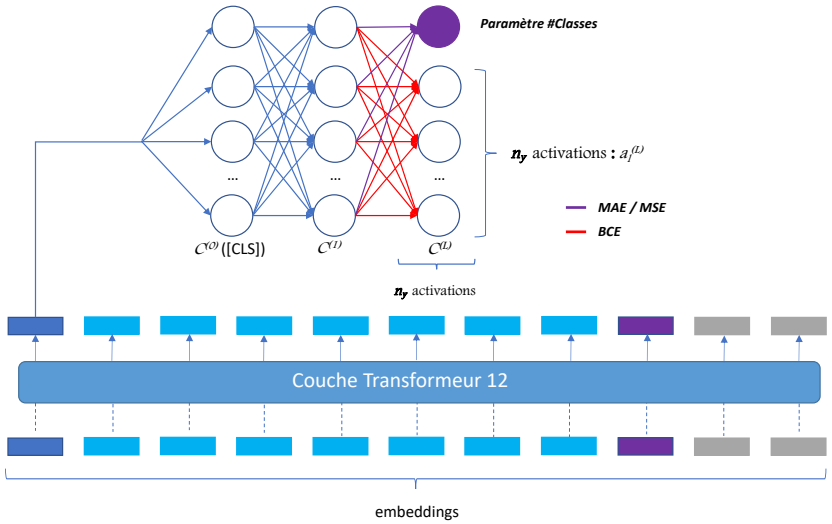

Figure 2: Architecture couche de classification dense avec un paramètre pour le calcul du nombre de classes présentes.

Le token [CLS], la sortie finale des couches transformeurs, contient une représentation du texte riche en informations. L'objectif de cette architecture est de pouvoir extraire à partir de ce token, et de l'état caché de la phrase qu'il contient, des critères ou des informations sur le nombre de sujets distincts présents dans le texte. Des critères qui serviront au calcul du paramètre.

4.2 Couche de seuillage (Threshold Layer TL)

La deuxième approche proposée se caractérise par l'ajout d'une couche dense à l'aval du classifieur, pour un total de $L = 3$ couches, ayant n_y neurones pour correspondre à la dernière couche de classification du modèle. Cet ajout a pour but de faire rapprocher les valeurs des activations finales le plus possible de 1 dans le cas de la présence du label, ou d'une valeur nulle dans le cas contraire. Par conséquent, le seuil trivial de 0, 5 peut être utilisé comme seuil pour la classification.

L'ajout de cette couche pourrait avoir comme effet l'augmentation du rappel du classifieur car plusieurs activations des labels non présents ne dépasseront plus le seuil de classification défini du fait qu'elle se rapprocheront le plus possible de 0. Un gain en précision peut aussi être espéré du fait que les activations des labels présents seront renforcées et mises plus en avant par rapport aux activations des labels non pertinents.

Pour cette approche, nous utilisons deux optimiseurs, un premier qui englobe et optimise toutes les couches du transformeurs ainsi que les deux premières couches du classifieur, et un autre dédié à l'optimisation de la dernière couche que l'on a rajoutée. La fonction d'erreur utilisée est la même pour les deux parties du classifieur, la BCE en l'occurrence. L'architecture finale du classifieur est présentée dans la figure 3

5 EXPÉRIMENTATIONS

Nous présentons dans cette section les résultats de l'évaluation de plusieurs modèles de langue transformeurs, principalement BERT et ses variantes, sur trois jeux de données de textes multilabels, dont le jeu de données français "HAL-Dataset" que nous avons conçu et mis à disposition. Nous comparons les différentes méthodes citées, ie. les méthodes de seuillage ainsi que les alternatives proposées à des approches de références, le tout mis en perspective avec des résultats cibles optimaux (approches oracles).

5.1 Modèles de langue évalués

Pour l'évaluation des méthodes proposées, nous utilisons la version *base* de **BERT**, avec 12 couches transformeurs et une taille du vecteur d'embeddings de 768, ainsi que quelques-unes de ses variantes :

- **RoBERTa** [Liu et al. 2019] une variante avec un processus d'entraînement optimisé, caractérisé par la suppression de la partie *prédiction de la phrase suivante* (ou *Next Sentence Prediction*) de BERT. Le jeu de données utilisé pour l'entraînement de RoBERTa est dix fois plus grand que celui utilisé pour BERT. Ceci a contribué à un gain en performance de RoBERTa par rapport à la version originale sur les tâches de traitement automatique du langage dans le comparatif GLUE [3];
- **DistilBERT**, une autre variante de BERT optimisée par l'utilisation de la distillation des connaissances pour avoir une approximation de BERT avec 40% moins de paramètres, tout en gardant 97% de ses performances. L'idée pour DistilBERT est basée sur le fait qu'après l'entraînement d'un modèle très grand, la distribution de la sortie peut être approximée par un réseau de neurones beaucoup plus petit, en utilisant la divergence Kulback-Leiber [Kullback and Leibler 1951] comme fonction d'optimisation [Sanh et al. 2020];
- **DeBERTa**, la variante la plus récente de BERT où les mots sont représentés par deux vecteurs qui encodent leur contenu et leur position relative dans la phrase. Il est aussi caractérisé

[3] https://gluebenchmark.com/tasks

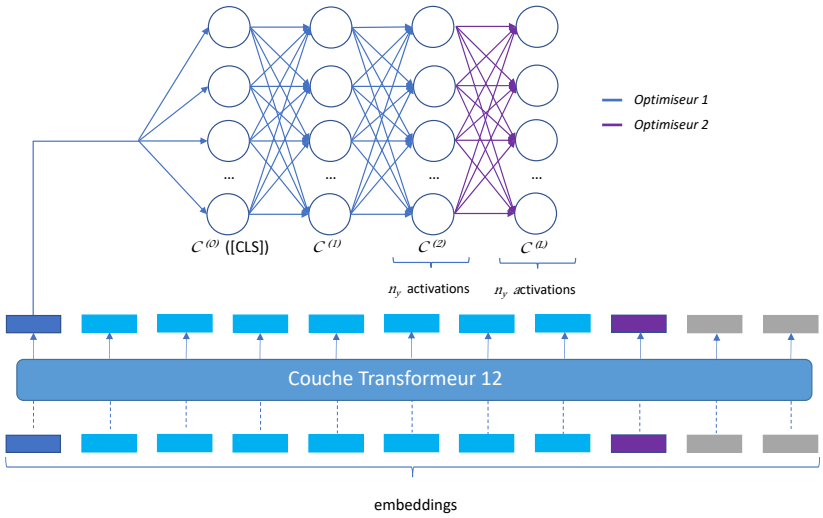

Figure 3: Architecture couche de classification dense de seuillage rajoutée à la fin du classifieur.

par l'optimisation du décodage de la prédiction des tokens masqués, ce qui contribue à un gain significatif en efficience dans la phase de pré-entraînement du modèle, mais aussi dans les performances concernant les différentes tâches de traitement du langage naturel. [He et al. 2021],

- **CamemBERT** est une variante basée sur RoBERTa [Martin et al. 2020], entraînée sur la partie française du corpus OSCAR [Suárez et al. 2019];
- **FlauBERT** [Le et al. 2020] variante entraînée sur divers sous-corpus français de différents styles d'écritures, allant des écritures formelles (par ex. Wikipedia et livres), aux écrits extraits d'internet (par ex. Common Crawl).

La comparaison des différents modèles de langue nous semble intéressante pour pouvoir évaluer les performances de nos approches ainsi que leur réutilisabilité et leur transposabilité sur différentes architectures de réseaux de neurones.

5.2 Corpus de textes utilisés

Comme nous l'avons déjà signalé, aucun LeaderBoard dédié à cette tâche n'existe. On retrouve peu de corpus multilabels utilisés d'une manière fréquente dans la littérature pouvant servir à l'évaluation et la comparaison des modèles, et ceci d'autant plus pour la langue française. Pour y remédier, nous avons constitué notre propre corpus de résumés d'articles scientifiques écrits en français à partir de HAL, corpus que nous mettons à disposition de tous.

Dans cette section, nous fournissons des détails concernant les différents jeu de données utilisés [4] pour l'évaluation des différents modèles :

- **HAL-Dataset** est un jeu de données en français que nous avons extrait de la plateforme d'archives ouvertes "HAL". Il contient les résumés de publications scientifiques de trois domaines (mathématiques, physique et informatique), publiées entre 1950 et 2020. L'extraction de ces résumés a été faite grâce à l'outil mis en disposition par HAL [5], en ne gardant que les articles pour lesquels le champs "résumé" français est présent. Ce qui représente 12 430 documents répartis sur 194 classes différentes, ici découpés en jeux d'entraînement, de validation et de test;
- **Reuter-21578** [6] est une collection d'articles du fil d'actualités de Reuters de l'année 1987 et publiée à l'année 1990. C'est un jeu de données qui a beaucoup été utilisé pour évaluer les modèles pour la CMLT de textes. Un article peut appartenir à un ou plusieurs domaines parmi 90;
- **AAPD** (ou ArXiv Academic Paper Dataset) est, comme pour le jeu de données "HAL-Dataset", une collection de la section "Résumé" (abstract) de plusieurs publications scientifiques. Un article scientifique peut avoir une ou plusieurs classifications parmi 54 classes. On utilise la même répartition entraînement, validation et test que [Yang et al. 2018].

Le tableau 1) et la figure 4 présentent plus en détail les différentes caractéristiques de ces jeux de données.

5.3 Méthode d'évaluation

Nous allons comparer les deux approches proposées ainsi que l'application des méthodes de seuillage aux transformeurs à des méthodes plus transparentes sur les critères de sélection, mais aussi avec des adaptations des modèles d'apprentissage profond à la classification multilabels de texte. Nous proposons aussi de comparer l'ensemble de ces méthodes à plusieurs bornes supérieures qui représentent les résultats optimaux que l'on peut atteindre.

5.3.1 Approches de références. Nous proposons dans un premier temps une comparaison avec des approches non-neuronales, avec

[4] Tous les jeux de données sont téléchargeables ici : https://github.com/hf-lis/Multi-Label-Datasets

[5] API HAL : https://api.archives-ouvertes.fr/docs/search

[6] https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection

Table 1: Les jeux de données avec W le nombre moyen de mots par document.

	#Train	#Valid	#Test	labels	W	#Train Epochs
Reuter-21578	5827	1943	3019	90	127,76	120
AAPD	53840	1000	1000	54	163,16	40
HAL-Dataset	9944	1243	1243	194	112,72	50

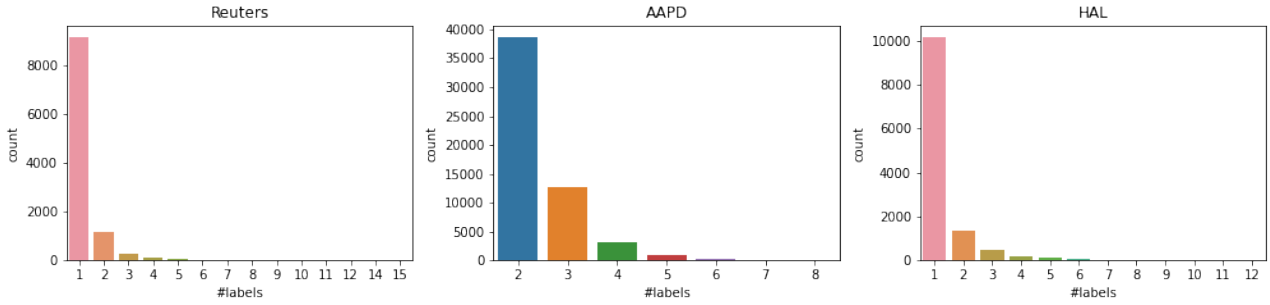

Figure 4: Compte des instances en fonction du nombre de labels pour l'ensemble des corpus.

des critères de classification interprétables, d'apprentissage machine, en utilisant les pondérations TF-IDF des documents comme entrées :

- Les arbres de décision en utilisant le critère *"Gini"* et sans imposer une profondeur maximale des arbres. Le nombre minimal utilisé pour les échantillons est de 2;
- Random Forest avec les mêmes paramètres utilisés dans la méthode précédente;
- le Bagging en utilisant les arbres de décision comme estimateur prinicipal (au nombre de 10);
- le GradientBoosting avec la régression logistique comme fonction d'erreur et une cadence d'apprentissage de 0,1;
- Support Vector machine (SVM) en utilisant RBF comme noyau et un paramètre de régularisation de 1,0.

Nous prenons aussi des approches neuronales comme éléments de comparaison pour l'évaluation de nos approches :

- **CNN** [Kim 2014] et **CNN-RNN** [Chen et al. 2017] qui utilisent des réseaux de neurones à convolutions pour extraire les caractéristiques propres au texte;
- **SGM** [Yang et al. 2018] qui applique un modèle de génération de séquences avec une nouvelle structure de décodeur pour résoudre le problème de CMLT;
- **MAGNET** [Pal et al. 2020] un réseau de graphes implémentant le mécanisme d'attention pour saisir la structure de dépendance entre les labels;
- **DocBERT** [Adhikari et al. 2019] : un fine-tuning des versions *base* et *large* de BERT pour la classification de documents.

5.3.2 L'optimal à atteindre. Les optimaux théoriques cibles (Approches oracles) se résument par les deux approches suivantes :

- L'approche oracle pour les N plus grandes activation, où nous considérons le nombre de labels présents pour une instance comme étant une donnée, pour ensuite prendre les N plus grandes activations comme labels présents;
- L'approche oracle pour les deux méthodes SCut de seuillage, où le calcul du seuil global et les seuils individuels est fait à partir du jeu de données de test. Cela pour avoir les résultats optimaux vers lesquels les deux méthodes de seuillage doivent s'approcher le plus possible.

5.4 Résultats

Dans cette partie, nous présentons les performances de toutes les approches citées dans les sections précédentes, testées sur les différents transformeurs présentés dans la section 5.1. Nous proposons les notations suivantes pour ces approches :

- "SGO" pour la méthode du Seuil Global Optimal (cf. section 3.2.1);00
- "SI" pour la méthode Seuil individuels (cf. section 3.2.2);
- "NPA" pour désigner l'approche des N plus grandes activations (cf. section 4.1);
- "TL" pour dénoter l'approche "couche de seuillage" (cf. section 4.2).

Les approches oracles quant à elles seront désignées en ajoutant $oracle$ à la suite des approches concernées : SGO_{oracle}, SI_{oracle} et NPA_{oracle}.

La longueur maximale des séquences considérée est de 512 tokens ainsi qu'un *batch size* de 8, et ce pour tous les jeux de données. Les tableaux 2 et 4 présentent les scores de micro-F1 (avec précision et rappel) et Accuracy pour le jeu de données de test des corpus en anglais et français respectivement. Les tableaux 3 et 5 présentent les optimaux à atteindre des différentes approches pour les corpus en anglais et celui en français. Les résultats des méthodes de seuillage ainsi que les architectures proposées ont été obtenus à partir des versions *base* des transformeurs utilisés, qui diffère de la version *large* de part le nombre des couches transformeurs (12 pour

Table 2: Scores pour les données de test issues des corpus en anglais Reuters et AAPD (les meilleurs scores sont en bleu gras).

Modèles	Reuters				AAPD			
	Pr.	R	F1	Acc	Pr.	R	F1	Acc
Decision Tree	78,36	75,05	76,67	74,23	49,67	46,8	48,19	26,6
Bagging	88,12	79,65	83,67	73,34	77,27	48,16	59,34	26,9
Random Forest	**97,18**	57,13	71,96	64,06	**94,2**	25,49	40,12	20
GradientBoost	88,06	80,56	84,14	74,23	79,73	46,8	58,98	27,1
SVM	94,19	79,62	86,29	80,64	80,85	59,98	68,86	36,2
CNN	-	-	86,3	-	-	-	66,4	-
CNN-RNN	-	-	85,5	-	-	-	66,9	-
SGM	-	-	-	-	-	-	71,0	-
MAGNET	-	-	89,9	-	-	-	69,6	-
DocBERT$_{base}$	-	-	89,0	-	-	-	73,4	-
DocBERT$_{large}$	-	-	90,7	-	-	-	**75,2**	-
Méthodes de seuillage, sans architecture spécifique (SGO et SI)								
BERT+*SGO*	90,0	91,74	90,86	86,45	75,56	**72,65**	74,08	41,7
BERT+*SI*	88,89	**92,30**	90,56	85,72	75,51	72,61	74,04	41,3
DistilBERT+*SGO*	90,83	90,78	90,80	86,29	75,73	72,08	73,86	39,8
DistilBERT+*SI*	88,40	91,66	90,0	85,29	79,06	68,31	73,3	41,01
RoBERTa+*SGO*	90,73	89,90	90,31	86,28	74,49	72,49	73,47	40,3
RoBERTa+*SI*	89,77	90,49	90,13	85,92	75,35	71,09	73,15	40,2
DeBERTa+*SGO*	91,63	90,41	**91,02**	**86,78**	74,46	73,56	74,01	39,4
DeBERTa+*SI*	90,67	90,92	90,79	86,61	75,87	72,08	73,92	40,7
Adaptation architecture Transformeurs (NPA et TL)								
BERT+*NPA*	92,33	85,87	88,98	85,92	73,48	66,38	69,75	40,3
BERT+*TL*	90,60	90,41	90,50	86,12	73,48	72,20	72,83	39,5
DistilBERT+*NPA*	92,08	86,03	88,95	86,15	73,93	66,29	69,90	41,8
DistilBERT+*TL*	90,0	89,66	89,83	85,89	73,63	72,32	72,97	40,2
RoBERTa+*NPA*	89,43	83,20	86,20	83,40	75,04	66,58	70,56	42,0
RoBERTa+*TL*	91,17	89,05	90,09	86,02	76,48	71,58	73,94	**42,5**
DeBERTa+*NPA*	92,22	86,41	89,21	86,35	75,32	65,67	70,16	41,9
DeBERTa+*TL*	90,97	90,43	90,70	86,12	75,97	71,70	73,78	41,8

Table 3: Scores des approches oracles pour les données de test des corpus en anglais Reuters et AAPD (les meilleurs scores sont en bleu gras).

Modèles	Reuters				AAPD			
	Pr.	R	F1	Acc	Pr.	R	F1	Acc
BERT+*SGO*$_{oracle}$	91,46	90,70	91,08	87,01	80,41	68,85	74,18	43,1
BERT+*SI*$_{oracle}$	93,61	91,24	92,41	87,94	**83,0**	70,59	**76,29**	44,9
DistilBERT+*SGO*$_{oracle}$	91,61	90,17	90,88	86,52	74,50	73,27	73,88	40,0
DistilBERT+*SI*$_{oracle}$	92,48	92,04	92,26	87,45	82,59	70,34	75,97	44,0
RoBERTa+*SGO*$_{oracle}$	91,21	89,56	90,38	86,32	74,12	71,83	72,96	39,9
RoBERTa+*SI*$_{oracle}$	93,18	90,57	91,86	87,78	81,95	70,14	75,58	44,2
DeBERTa+*SGO*$_{oracle}$	92,74	90,73	91,72	87,41	75,21	71,95	73,55	40,4
DeBERTa+*SI*$_{oracle}$	**93,74**	91,56	**92,64**	88,27	82,82	70,26	76,02	44,0
BERT+*NPA*$_{oracle}$	92,55	**92,55**	92,55	**92,45**	74,06	74,06	74,06	51,8
DistilBERT+*NPA*$_{oracle}$	91,99	91,99	91,99	91,95	75,09	**75,09**	75,09	**54,7**
RoBERTa +*NPA*$_{oracle}$	91,69	91,69	91,69	91,45	73,36	73,36	73,36	51,5
DeBERTa +*NPA*$_{oracle}$	92,31	92,31	92,31	92,41	73,48	73,48	73,48	52,7

Table 4: Scores pour les données de test issues du corpus en français HAL-Dataset (les meilleurs scores sont en bleu gras).

Modèles	HAL-Dataset			
	Pr.	R	F1	Acc
Decision Tree	56,96	52,42	54,60	62,27
Bagging	86,24	52,66	65,39	62,35
Random Forest	90,72	50,85	65,17	64,60
GradientBoost	64,14	57,32	60,54	60,58
SVM	**95,23**	55,56	70,18	66,37
Seuillage sans architecture spécifique (SGO et SI)				
CamemBERT+*SGO*	79,17	67,11	72,64	71,84
CamemBERT+*SI*	77,12	67,65	72,08	71,35
FlauBERT+*SGO*	80,09	69,58	**74,47**	73,12
FlauBERT+*SI*	78,61	**70,01**	74,06	**73,45**
Adaptation architecture Transformeurs (NPA et TL)				
CamemBERT+*NPA*	66,45	49,93	57,02	64,36
CamemBERT+*TL*	88,38	59,31	70,98	68,78
FlauBERT+*NPA*	65,70	60,58	63,04	67,42
FlauBERT+*TL*	82,49	66,08	73,38	71,6

Table 5: Scores des approches oracles pour les données de test du corpus en français HAL-Dataset(les meilleurs scores sont en bleu gras).

Modèles	HAL-Dataset			
	Pr.	R	F1	Acc
CamemBERT+SGO_{oracle}	83,20	65,59	73,36	71,92
CamemBERT+SI_{oracle}	**90,33**	67,83	77,48	72,96
FlauBERT+SGO_{oracle}	83,16	68,38	75,05	73,61
FlauBERT+SI_{oracle}	89,21	70,49	**78,75**	75,30
CamemBERT+NPA_{oracle}	70,80	70,80	70,80	75,46
FlauBERT+NPA_{oracle}	72,85	**72,85**	72,85	**76,75**

la version base contre 24 pour la version large), ainsi que la taille du vecteur d'embedding (768 pour la version base contre 1024 pour la version large). L'entraînement des modèles a été réalisé à l'aide d'une Nvidia RTX 2080 Ti (11 Go de mémoire vidéo), le tableau 6 présente les différentes durées des phases d'entraînement et de validation pour les jeux de données étudiés.

Les modèles BERT et dérivés surpassent toutes les autres méthodes, que ce soient les méthodes classiques comme les SVM ou le GradientBoost, ou les autres méthodes d'apprentissage profond. Les méthodes de seuillage (*SGO* et *SI*) ainsi que les architectures proposées (*NPA* et *TL*) dépassent la version *base* de *DocBERT* (état de l'art actuel de la classification multilabels de textes pour les corpus AAPD et Reuters), et dans certains cas sa version *large*. L'implantation des méthodes *SGO* et *SI* sont les plus performantes parmi toutes les approches étudiées, avec un score micro-F1 de **91,02** et **90.79** respectivement, comparés au score de **90,7** de la version large de *DocBERT* pour le corpus *Reuters* (cf. tableau 2).

Table 6: Durée (en secondes) d'une itération dans la phase d'entraînement et de l'inférence pour la phase de validation des modèle BERT et FlauBERT sur les jeux de donnés AAPD, Reuters et HAL avec un batch size de 8

Modèles	AAPD		Reuters	
	Train	Valid	Train	Valid
BERT+*SGO*	2822	13,5	315	26,4
BERT+*SI*	3202	13,5	420	26,5
BERT+*NPA*	2031	13,7	223	27
BERT+*TL*	2015	25	213	50,8
	HAL			
	Train	Valid		
FlauBERT+*SGO*	539	15.6		
FlauBERT+*SI*	836	15.7		
FlauBERT+*NPA*	369	16.1		
FlauBERT+*TL*	379	29.6		

Ces deux méthodes améliorent de façon notable le score micro F1 et l'exactitude (*Accuracy*) des modèles, mais sont celles qui ont les coûts d'entraînement les plus élevés. Ceci est dû au fait que le calcul des seuils dans la phase d'entraînement requiert un temps de traitement considérable (cf. tableau 6). Le(s) seuil(s) de détection baisse parfois, ce qui a pour effet d'augmenter le nombre de vrais positifs; dans d'autre cas, ce(s) seuil(s) augmente, ce qui diminue le nombre de faux positifs. La précision du modèle augmente ainsi que la micro-F1 et l'exactitude.

Pour ce même corpus, l'architecture *TL* dépasse la version *base* du modèle *DocBERT* ainsi que le modèle *MAGNET*, et se rapproche de la version *large* du premier avec un score de **90,60** pour le modèle *DeBERTa*. L'approche *NPA* quant à elle ne réussit pas à atteindre son optimal théorique, en obtenant un score quand bien même dépassant la version *base* de *DocBERT* (cf. partie Adaptation architecture Transformeur du tableau 2). Ces deux architectures nécessitent un temps d'entraînement moins important que les deux méthodes de seuillage (une différence de 1000 secondes dans la configuration de nos tests pour le modèle BERT) , mais possèdent un coût d'inférence plus élevé, notamment pour l'architecture *TL* où le temps d'inférence, **25** et **29.6** secondes pour BERT et FlauBERT respectivement, est presque deux fois celui des autres méthodes (cf. tableau 6). Cela peut être expliqué par la présence de la couche supplémentaire que possède cet architecture, augmentant ainsi le temps d'inférence des modèles, chose qui n'impacte pas de façon considérable la phase d'entraînement de par sa nature parallèle (l'inférence étant exécutée d'une manière séquentielle couche après couche).

Pour le corpus AAPD, les méthodes de seuillage ainsi que l'approche *TL* réussissent à obtenir de bonnes performances comparées aux autres méthodes, mais le modèle $DocBERT_{large}$ reste le plus performant parmi toutes les approches étudiées. La nature complexe du vocabulaire scientifique de ce corpus souligne l'apport que peut avoir l'augmentation de la taille des modèles transformeurs (rajout de plusieurs couches et augmentation de la taille des embeddings).

L'approche SI_{oracle} est la plus performante des approches oracles étudiées. En effet, le calcul d'un seuil individuel pour chaque label à partir du jeu de test conduit vers un optimal général pour toutes les

classes qui dépassent l'optimal de la méthode *SGO*. Mais cela n'est pas le cas pour son équivalent expérimental. *SGO* reste la méthode qui s'approche le plus de son optimal théorique (cf. tableaux 3 et 2). Chose qui peut être expliquée par le fait que pour la méthode *SI* plusieurs seuils doivent être calculés pour chaque label à partir du jeu d'entraînement, ce qui ne garantit par un optimal général sur l'ensemble des classes pour le jeu de test. Cet effet est accentué si le nombre de classes est grand.

Pour l'optimal théorique de la méthode *NPA*, fournir le nombre effectif de labels pertinents pour une instance permet de considérer présents de nouveaux labels, ce qui augmente le taux de vrais positifs et baisse celui des faux négatifs. Mais cela a pour effet d'augmenter aussi le risque de faux positifs car ces nouveaux labels sont dans certains cas des prédictions non valides. Les instances pour lesquelles cette approche réduit le nombre de labels précédemment prédits sont rares. Donc le score micro F1 reste inchangé ou peut baisser par rapport aux optimaux théoriques des approches de seuillage. D'autre part, une augmentation considérable de l'exactitude est constatée, c'est d'ailleurs l'approche qui réussit à obtenir les scores d'exactitude les plus élevés (cf. tableau 3 et 5). L'implantation effective de cette approche quant à elle n'a pas permis d'obtenir les même performances que son optimal théorique. L'état caché de la phrase contenu dans le token [CLS] n'est peut-être pas suffisant pour le calcul du nombre de classes présentes. Dans le futur, l'exploitation des scores d'attention obtenus dans les différentes couches du modèle pourrait être une méthode plus efficace pour accomplir cette tâche.

On note aussi que les deux architectures proposées obtiennent des scores de micro-précision plus élevés que les approches de seuillage. Pour l'approche *NPA*, cela est dû au fait que la sélection des labels est réalisée suivant l'approximation du nombre de classes calculé. Pour *TL* cela peut être dû au fait que les valeurs des activations des labels considérés comme présents sont accentuées, contre l'atténuation des labels considérés comme non pertinents.

Les mêmes tendances sont observées pour notre corpus de texte en français *"HAL-Dataset"*. Les méthodes de seuillage obtiennent les scores micro-F1 les plus élevés, **74,47** pour *SGO* et **74,06** pour *SI*, (cf. tableau 4) suivies de l'architecture *TL* avec un score de **73,38**. Un score proche des méthodes précédentes avec une architecture qui ne requiert pas une phase de recherche de seuils optimaux.

On note tout de même que l'approche *NPA*, que ce soit l'optimal théorique ou l'implantation expérimentale, n'est pas aussi performantes que les autres méthodes. Cela pourrait être dû à la difficulté de l'apprentissage du nombre de label compte tenu du déséquilibre du jeu de données. Environ 82% des instances n'ont qu'un seul label (cf. figure 4).

DeBERTa est la variante de *BERT* la plus performante parmi tous les modèles évalués, les changements qu'apportent cette variante compte tenu de l'ajout de l'information concernant la position relative du mot par rapport à la phrase, semblent améliorer les performances de *BERT*. Cette amélioration du rendement vient au détriment de la vitesse d'entraînement et d'inférence du modèle qui devient plus lent que les autres variantes. *DistilBERT* quant à lui obtient des résultats au même niveau que sa version originale, malgré sa taille réduite (moins de couches transformeurs). La distillation des connaissances semble être une méthode efficace pour pallier à l'inconvénient majeur des transformeurs et des réseaux neuronaux : la nécessité d'un temps d'entraînement très élevé.

Pour la tâche de classification multilabels de texte, *RoBERTa* n'est pas aussi performant que les autres variantes. Cela peut être dû à l'absence de la partie "prédiction de la phrase suivante", une partie qui peut avoir une importance pour la classification multilabels, dans la mesure où elle peut permettre au modèle d'apprendre la manière dans les idées d'un même domaine se succède et de pouvoir faire la différence entre plusieurs champs lexicaux.

CamemBERT qui est une variante basée sur *RoBERTa*, hérite des mêmes problématiques face à la classification multilabels, et se voit dépassée par *FlauBERT* dont la force réside dans le fait qu'il soit entraîné sur des corpus très variés, avec des styles d'écritures diverses.

On note aussi que les SVM et les forêts d'arbres décisionnels obtiennent les scores de micro-précision les plus élevés, au détriment du taux de rappel, faisant ainsi baisser le score micro F1. Mais la précision n'est pas un facteur suffisant pour la mesure des performances. SVM peut être considéré comme l'approche non neuronale la plus performante en vu de son score d'exactitude élevé, mais qui reste en dessous des autre méthodes testées.

6 CONCLUSION ET TRAVAUX FUTURS

Les modèles de langue à base de transformeurs surpassent les autres architectures neuronales profondes et constituent une base solide adaptable pour une multitude de tâches de traitement automatique des langues et la classification de textes, c'est la direction que nous avons suivie dans cette étude.

Tout d'abord, nous avons testé et montré dans cet article que les approches de seuillage sont très performantes pour la classification multilabels. Le calcul d'un seuil global obtient des résultats plus élevés que le calcul d'un seuil individuel pour chaque classe. L'optimisation faite sur chaque classe ne garantit pas un optimal général pour tous les labels. Nous avons par la suite proposé des modifications sur l'architecture des transformeurs. Ceci par le rajout d'une couche supplémentaire, avec un nombre d'activations égal au nombre de classes, à l'aval du modèle pour s'affranchir d'une optimisation sur les seuils. Une approche qui, en moyenne, est aussi performante que les approches de seuillage. L'étude des optimaux à atteindre a montré que l'utilisation du nombre de classes pour la sélection des labels pertinents augmente de façon considérable les performances de la classification multilabels. Cependant, la différence entre ces optimaux et les résultats effectifs de nos expérimentations montrent que notre proposition doit être améliorée dans le cas des jeux de données déséquilibrés. Au final les deux familles d'approches, seuillage et modification de l'architecture des transformeurs, sont des méthodes qui améliorent les performances des réseaux d'apprentissage profond pour la classification multilabels de texte.

La langue des corpus de texte, anglais pour AAPD et Reuters, ou français pour le corpus HAL-Dataset que nous avons construit, ainsi que leur nature (article scientifiques ou dépêches journalistiques) ne semblent pas être des facteurs qui impactent les performances des approches proposées. Ce sont d'ailleurs des approches qui peuvent être utilisées pour tout problème de classification multilabels. Dans le contexte applicatif de nos travaux de thèse, nous construisons un

jeu de données à partir des mails clients de l'agence, et sur lequel nous testerons toutes ces approches.

Chaque variante de *BERT* cherche à améliorer les aspects contraignants de ce modèle. *DistilBERT*, malgré sa taille réduite, obtient des résultats aussi élevés que la version originale. *DeBERTa* se positionne comme étant la variante la plus performante, en compagnie de *FlauBERT* qui dépasse *CamemBERT*, la version basée sur *RoBERTa* dont les performances restent derrière les autres variantes.

Nous avons montré l'intérêt réel que peuvent avoir les approches d'exploitation des activations des couches de sorties. D'autres méthodes d'adaptation des réseaux neuronaux pour la classification multilabels peuvent être explorées, dans la mesure où les seuils de détection peuvent être des paramètres appris du modèle lors de la phase d'entraînement. Des améliorations peuvent être faites sur les architectures alternatives au seuillage proposées, notamment la recherche d'une méthode plus efficace pour le calcul du nombre de classes présentes pour la méthode *NPA* ainsi que l'augmentation du nombre de couches de seuillage pour l'approche *TL*. Toutes ces approches peuvent être utilisées pour l'optimisation de tout modèle d'apprentissage profond.

En ce qui concerne le domaine d'application, la classification multilabels est une tâche pertinente dans le monde industriel. Cependant la CMLT n'est pas une tâche si fréquente et on regrette qu'elle ne figure pas parmi les comparatifs les plus en vue comme GLUE.

REFERENCES

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. DocBERT: BERT for Document Classification. *arXiv:1904.08398 [cs]* (April 2019). http://arxiv.org/abs/1904.08398 arXiv: 1904.08398 version: 1.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]* (May 2016). http://arxiv.org/abs/1409.0473 arXiv: 1409.0473.

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. 2004. *Learning multi-label scene classification.*

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and I. Dhillon. 2019. X-BERT: eXtreme Multi-label Text Classification with using Bidirectional Encoder Representations from Transformers. *undefined* (2019). https://www.semanticscholar.org/paper/X-BERT%3A-eXtreme-Multi-label-Text-Classification-Chang-Yu/3b9efab2114a3456dccb9d625ca732100d18ba74

Guibin Chen, Deheng Ye, Zhenchang Xing, Jieshan Chen, and Erik Cambria. 2017. *Ensemble application of convolutional and recurrent neural networks for multi-label text categorization.* https://doi.org/10.1109/IJCNN.2017.7966144 Pages: 2383.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]* (May 2019). http://arxiv.org/abs/1810.04805 arXiv: 1810.04805.

Luyun Gan, Brosnan Yuen, and Tao Lu. 2019. Multi-label Classification with Optimal Thresholding for Multi-composition Spectroscopic Analysis. *arXiv:1906.10242 [cs, eess, stat]* (June 2019). http://arxiv.org/abs/1906.10242 arXiv: 1906.10242.

Jibing Gong, Zhiyong Teng, Qi Teng, Hekai Zhang, Linfeng Du, Shuai Chen, Md Zakirul Alam Bhuiyan, Jianhua Li, Mingsheng Liu, and Hongyuan Ma. 2020. Hierarchical Graph Transformer-Based Deep Learning Model for Large-Scale Multi-Label Text Classification. *IEEE Access* 8 (2020), 30885–30896. https://doi.org/10.1109/ACCESS.2020.2972751 Conference Name: IEEE Access.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *arXiv:2006.03654 [cs]* (Jan. 2021). http://arxiv.org/abs/2006.03654 arXiv: 2006.03654.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (Dec. 1997), 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *arXiv:1408.5882 [cs]* (Sept. 2014). http://arxiv.org/abs/1408.5882 arXiv: 1408.5882.

S. Kullback and R. A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22, 1 (March 1951), 79–86. https://doi.org/10.1214/aoms/1177729694 Publisher: Institute of Mathematical Statistics.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15).* AAAI Press, Austin, Texas, 2267–2273.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. 2020. FlauBERT: Unsupervised Language Model Pre-training for French. *arXiv:1912.05372 [cs]* (March 2020). http://arxiv.org/abs/1912.05372 arXiv: 1912.05372.

David Lewis, Ctr Info, Lang Studies, and Marc Ringuette. 1996. A Comparison of Two Learning Algorithms for Text Categorization. *Third Annual Symposium on Document Analysis and Information Retrieval* (Oct. 1996).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]* (July 2019). http://arxiv.org/abs/1907.11692 arXiv: 1907.11692 version: 1.

Oscar Luaces, Jorge Díez, José Barranquero, Juan José del Coz, and Antonio Bahamonde. 2012. Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence* 1, 4 (Dec. 2012), 303–313. https://doi.org/10.1007/s13748-012-0030-x Number: 4.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a Tasty French Language Model. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), 7203–7219. https://doi.org/10.18653/v1/2020.acl-main.645 arXiv: 1911.03894.

Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. *IEEE Transactions on Knowledge and Data Engineering* 18, 10 (Oct. 2006), 1338–1351. https://doi.org/10.1109/TKDE.2006.162 Number: 10 Conference Name: IEEE Transactions on Knowledge and Data Engineering.

Ankit Pal, Muru Selvakumar, and Malaikannan Sankarasubbu. 2020. Multi-Label Text Classification using Attention-based Graph Neural Network. *Proceedings of the 12th International Conference on Agents and Artificial Intelligence* (2020), 494–505. https://doi.org/10.5220/0008940304940505 arXiv: 2003.11644.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv:1802.05365 [cs]* (March 2018). http://arxiv.org/abs/1802.05365 arXiv: 1802.05365.

Alec Radford, Jeff Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019). /paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108 [cs]* (Feb. 2020). http://arxiv.org/abs/1910.01108 arXiv: 1910.01108.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215 [cs]* (Dec. 2014). http://arxiv.org/abs/1409.3215 arXiv: 1409.3215.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures. Leibniz-Institut für Deutsche Sprache. https://doi.org/10.14618/IDS-PUB-9021

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. Mining Multi-label Data. In *Data Mining and Knowledge Discovery Handbook*, Oded Maimon and Lior Rokach (Eds.). Springer US, Boston, MA, 667–685. https://doi.org/10.1007/978-0-387-09823-4_34

Grigorios Tsoumakas and Ioannis Vlahavas. 2007. Random k-Labelsets: An Ensemble Method for Multilabel Classification. In *Machine Learning: ECML 2007 (Lecture Notes in Computer Science)*, Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron (Eds.). Springer, Berlin, Heidelberg, 406–417. https://doi.org/10.1007/978-3-540-74958-5_38

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]* (Dec. 2017). http://arxiv.org/abs/1706.03762 arXiv: 1706.03762.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: Sequence Generation Model for Multi-label Classification. *arXiv:1806.04822 [cs]* (June 2018). http://arxiv.org/abs/1806.04822 arXiv: 1806.04822.

Yiming Yang. 1997. *An evaluation of statistical approach to text categorization.* Technical Report.

Yiming Yang. 2001. A study of thresholding strategies for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '01).* Association for Computing Machinery, New York, NY, USA, 137–145. https://doi.org/10.1145/383952.383975

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics, San Diego, California, 1480–1489. https://doi.org/10.18653/v1/N16-1174

Min-Ling Zhang and Zhi-Hua Zhou. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition* 40, 7 (July 2007), 2038–2048. https://doi.org/10.1016/j.patcog.2006.12.019 Number: 7.