Classification non supervisée de documents à partir des modèles Transformeurs

Mira Ait-Saada[*,**], François Role[*], Mohamed Nadif[*]

[*]Université de Paris, CNRS, Centre Borelli UMR9010, 75006 Paris
[**]Caisse des Dépôts et Consignations, Datalab, 75013, Paris
<prénom.nom>@u-paris.fr

Résumé. Les plongements de mots pré-entraînés basés sur les modèles Transformeur sont maintenant largement utilisés dans l'exploration de textes où ils sont connus pour améliorer de manière significative les tâches supervisées telles que la classification supervisée de textes, la reconnaissance d'entités nommées et la réponse aux questions. Puisque les modèles Transformeur créent plusieurs plongements différents pour la même entrée, un à chaque couche de leur architecture, diverses études ont déjà essayé d'identifier ceux de ces plongements qui contribuent le plus au succès des tâches mentionnées ci-dessus. En revanche, la même analyse des performances n'a pas encore été réalisée dans le cadre non supervisé. Dans cet article, nous évaluons l'efficacité des modèles Transformeur sur l'importante de la tâche de classification non supervisée de documents. En particulier, nous présentons une approche *clustering ensemble* qui exploite toutes les couches du réseau. Des expériences numériques menées sur des ensembles de données réels avec différents modèles Transformeur montrent l'efficacité de la méthode proposée par rapport à plusieurs stratégies habituellement utilisées. Cet article est une restitution plus détaillée du papier (Ait-Saada et al., 2021a).

1 Introduction

À partir de BERT (Devlin et al., 2019), les plongements de mots contextualisés fournis par les modèles de langage neuronaux ont été de plus en plus utilisés comme entrée de nombreuses applications de traitement automatique de la langue (TAL), où ils contribuent grandement à atteindre des niveaux de performances impressionnants. Depuis, un nombre important de travaux se sont intéressés à disséquer ces modèles multi-couches afin de mettre la lumière sur ces boîtes noires Kovaleva et al. (2019); Ait-Saada et al. (2021b); Clark et al. (2019).

Un modèle Transformeur produit plusieurs représentations pour chaque mot (une à chaque couche de l'architecture du réseau) et des études dans le domaine de l'apprentissage supervisé ont tenté de déterminer le type d'information capturé par les différentes couches. Par exemple, dans (Peters et al., 2018b; Tenney et al., 2019), en utilisant des plongements issus de modèles Transformeur pré-entraînés en entrée d'une suite de tâches de TAL, les auteurs sont d'accord pour dire que les premières couches encodent la plupart des phénomènes syntaxiques locaux tandis que des aspects sémantiques plus complexes apparaissent aux couches supérieures.

D'autre part, s'intéressant plus spécifiquement aux capacités de généralisation des plongements de mots contextualisés (comprenant les algorithmes ELMo, BERT et OpenAI *Transformer*), Liu et al. (2019a) ont observé que les couches intermédiaires des Transformeurs présentent une meilleure transférabilité, tandis que Hao et al. (2019) ont observé que les premières couches de BERT-large sont plus stables et varient moins d'une tâche à l'autre lorsque le modèle est ré-entraîné sur différentes tâches et en ont conclu qu'elles étaient plus transférables que les couches supérieures. Dans un autre axe de recherche, certaines études se sont concentrées sur l'impact du *fine-tuning* ou ré-entrainement des modèles Transformeurs, et ont vérifié expérimentalement que plus on se rapproche de la dernière couche, plus les représentations deviennent spécifiques à la tâche apprise par le modèle (van Aken et al., 2019; Kovaleva et al., 2019).

Le principal point à retenir de ces études est que les représentations fournies par les différentes couches capturent clairement des informations différentes, conduisant ainsi à des résultats très différents lorsqu'elles sont utilisées comme entrée d'une tâche de *text-mining*. Le problème est qu'il n'est pas possible de savoir à l'avance laquelle de ces couches sera la plus à même de donner les meilleurs résultats pour une tâche donnée, notamment dans un contexte non-supervisé, comme le nôtre. Lors de l'utilisation de plongements pré-entraînés, une règle empirique courante consiste à exclure la dernière couche en supposant qu'elle est biaisée par rapport aux cibles d'entraînement, comme démontré par van Aken et al. (2019). Les toutes premières couches sont également exclues en général, car elles sont jugées trop proches de l'entrée du modèle. Une autre approche pour sélectionner une couche afin d'effectuer une tâche donnée consiste à utiliser un ensemble de données étiqueté (ou labellisé) comme ensemble de développement pour déterminer la meilleure couche à utiliser pour les nouveaux ensembles de données (Zhang et al., 2020). Nous montrons que cette approche n'est pas optimale dans notre cas, puisque la meilleure couche est souvent différente d'un jeu de données à l'autre.

De plus, nous soutenons que les caractéristiques sémantiques sont très utiles dans le regroupement de textes, et il a été observé dans (Tenney et al., 2019) que, contrairement aux informations syntaxiques, qui sont généralement concentrées sur quelques couches, les caractéristiques sémantiques sont réparties sur l'ensemble du réseau. C'est pourquoi, au lieu de choisir une couche unique, nous préférons exploiter toutes les représentations fournies par les modèles Transformeur pour effectuer un apprentissage non-supervisé. Pour y parvenir, nous proposons de partitionner séparément les représentations de documents calculées à chaque couche, puis d'en déduire une partition *consensuelle*, en exploitant toutes les informations fournies à chaque niveau du réseau profond. Afin d'évaluer notre approche, nous la comparons à des approches de base précédemment utilisées, incluant la concaténation et la moyenne des couches, l'utilisation de l'avant-dernière couche ainsi que la combinaison des quatre dernières couches. Nous comparons également nos résultats à ceux obtenus avec une représentation standard de sac de mots (ou *Bag-Of-Words*) afin de mesurer l'intérêt des Transformeurs pour la tâche de *clustering*.

Par ailleurs, nous étudions l'effet de la réduction de dimension sur les représentations issues de Transformeurs, un sujet qui a rarement été étudié jusqu'à présent. Pour un examen de cette question dans le contexte des plongements de mots antérieurs à BERT tels que Word2vec, fastText et GloVe, le lecteur est renvoyé à (Raunak et al., 2019) qui a montré qu'il était possible de réduire de moitié les dimensions vectorielles sans altérer de manière significative les performances. Une méthode de compression plus spécifique est proposée par Li et Eisner (2019) pour les représentations de mots ELMo (Peters et al., 2018a), qui vise à extraire les informations les

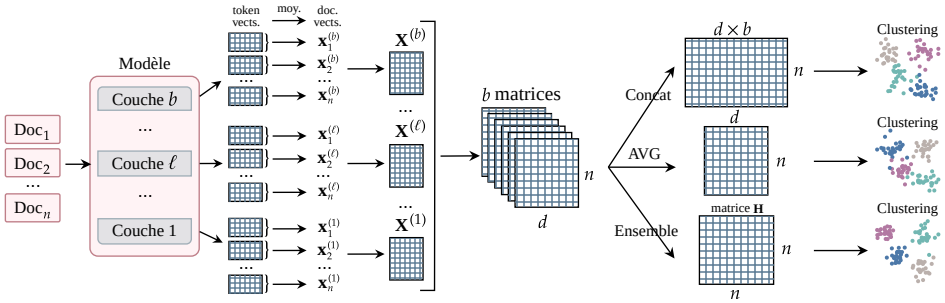

FIG. 1 – Description des différentes manières de combiner les couches ℓ d'un modèle Transformeur. $\mathbf{x}_i^{(\ell)}$ est la représentation du document calculé en faisant la moyenne des représentations (obtenues à la couche ℓ) des tokens contenus dans le document i. Ce vecteur forme la i-ième ligne de la matrice $\mathbf{X}^{(\ell)}$, qui est la représentation de l'ensemble de données à la couche ℓ.

plus utiles pour effectuer de l'analyse des dépendances. À notre connaissance, rien n'a été fait pour évaluer l'impact que peut avoir une réduction de dimensions sur les représentations issues de Transformeurs. Le présent article vise à contribuer à combler cette lacune dans le contexte des modèles Transformeur, en réduisant la dimensionnalité des représentations d'origine de manière encore plus drastique et néanmoins plus efficace.

La papier est organisé de la manière suivante. Dans la Section 2 nous montrons que le fait de combiner le *clustering ensemble* et la réduction de dimension permet d'augmenter significativement les performances de clustering sur plusieurs jeux de données réels. La section 3 met ensuite en évidence un avantage important du *clustering ensemble* que nous proposons, à savoir l'estimation efficace du nombre de classes, en même temps qu'un partitionnement efficace, ce qui est très utile lorsque le nombre exact de classes est inconnu. Enfin, la conclusion rappelle notre principale contribution et quelques perspectives à notre approche.

2 Classification non supervisée à partir des plongements issus de Transformeurs

Pour un jeu de données de n documents et un modèle Transformeur avec b couches, on obtient b différentes représentations denses de taille d, une pour chaque couche. Étant donné une couche ℓ, la représentation du i-ième document du jeu de données est obtenue à partir des plongements de ses 512 premiers tokens qui sont regroupés en utilisant une moyenne (comme suggéré dans (Xiao, 2018; Reimers et Gurevych, 2019)), obtenant ainsi un vecteur $\mathbf{x}_i^{(\ell)}$ de taille d, qui constitue la i-ième ligne de la matrice $\mathbf{X}^{(\ell)}$. Le jeu de données peut alors être représenté par b différentes matrices $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(b)}$ de taille $n \times d$ comme le montre la figure 1. Nous appelons « partition » le résultat fourni par l'une des exécutions de l'algorithme de *clustering*. Une partition contient n labels où le i-th label correspond à la classe à laquelle le i-th document du jeu de données est affecté.

Pour un modèle à b couches, on peut penser à exécuter un algorithme de *clustering* sur chacune des matrices $\mathbf{X}^{(\ell)}$, et choisir celle qui donne le « meilleur » résultat. Cependant, dans

le cadre non supervisé où aucun label n'est disponible, il n'y a pas de moyen simple de savoir quelle matrice $\mathbf{X}^{(\ell)}$ est susceptible de donner ce meilleur résultat. Ainsi, nous décrivons deux manières d'exploiter les différentes représentations fournies par un modèle Transformeur :

— En agrégeant les matrices $\mathbf{X}^{(\ell)}$, $\ell = 1, \dots b$ et en appliquant un *clustering* sur l'agrégat.

— En utilisant les matrices $\mathbf{X}^{(\ell)}$ individuellement dans le cadre d'une approche *ensemble*.

Nous évaluons également les performances obtenues en rajoutant une étape de réduction de dimension basée sur l'ACP, réduisant les représentations à seulement $d' = 100$ dimensions.

2.1 Post-traitement des plongements

L'utilisation de la réduction de dimension linéaire est courante en traitement automatique de la langue et a montré des améliorations prometteuses sur diverses représentations (Raunak et al., 2019; Mu et Viswanath, 2018; Xu et al., 2003). Dans notre étude, nous étudions le pouvoir des composantes principales (CP) dans la réduction de la dimension des représentations textuelles denses tout en préservant l'information qu'elles contiennent. En particulier, nous avons observé que l'utilisation des premières composantes principales n'altère pas les performances, même en compressant les vecteurs d'origine à 10% des dimensions. Plus important encore, nous montrons qu'une opération de *whitening* supplémentaire appliquée sur les composantes principales conduit à des améliorations surprenantes des performances de clustering tout en réduisant drastiquement la dimensionnalité. Elle consiste à construire une représentation réduite $\mathbf{Y}$ où chaque valeur est calculée comme suit :

$$y_{ij} = \mathbf{x}_i \mathbf{w}_j^\top / \sqrt{\sigma_j} \; ; \; i = 1, \dots, n \; ; \; j = 1, \dots, d'$$

où $\mathbf{x}_i$ est la ie ligne de $\mathbf{X}$, $\mathbf{w}_j$ est le je vecteur propre de $\mathbf{X}^\top \mathbf{X}$ et σ_j sa je valeur propre. L'opération de *whitening* peut être considérée comme une meilleure alternative à la suppression des composantes principales dominantes (Raunak et al., 2019; Mu et Viswanath, 2018), en normalisant plutôt les CP à la variance unitaire, réduisant ainsi l'impact des premières composantes et produisant des vecteurs de meilleure qualité. Dans ce qui suit, nous appelons le post-traitement basé sur l'ACP avec *whitening* ACP$_w$.

2.2 Agrégation des représentations multi-couches

Pour un document i, la première méthode de combinaison consiste à moyenner les b vecteurs $\mathbf{x}_i^{(\ell)}$, $\ell = 1, \dots b$, obtenant ainsi un vecteur unique de taille d, comme dans (Vulić et al., 2020), représentant la i-ème ligne d'une matrice de données de taille $n \times d$. Nous appelons cette méthode AVG. La deuxième approche, que nous appelons Concat, consiste à concaténer les b vecteurs $\mathbf{x}_i^{(\ell)}$, ce qui donne un vecteur unique de taille $b \times d$, comme effectué dans (Devlin et al., 2019) en utilisant les dernières couches. Ces deux manières de combiner les représentations issues des b couches sont également décrites par la figure 1. En plus de ces représentations agrégées, nous effectuons éventuellement une réduction de dimension basée sur l'ACP (ACP$_w$) avant d'exécuter un algorithme de *clustering*, obtenant des représentations de taille $d' = 100$.

2.3 Approche *ensemble*

Une autre façon de combiner les informations fournies par toutes les couches est, non pas de les agréger en amont, mais de s'appuyer sur une procédure *clustering ensemble* ou consensus, qu'on désigne par ENS (algorithme 1). L'approche *ensemble* a été considérée dans différents

Algorithme 1 : Clustering Ensemble

entrée : Un jeu de données $\mathcal{D}$; un modèle Transformeur $\mathcal{M}$ à b couches, deux algorithmes de *clustering* $\mathcal{C}_1$ et $\mathcal{C}_2$; le nombre de clusters k
sortie : Une partition consensuelle $\mathbf{p}^*$
1 **pour** $\ell = 1, \ldots, b$ **faire**
2 $\quad \mathbf{X}^{(\ell)} \leftarrow$ plongements de documents calculés avec $\mathcal{M}(\mathcal{D})$ à chaque couche ℓ (comme le montre la figure 1) ;
3 $\quad \mathbf{p}^{(\ell)} \leftarrow \mathcal{C}_1(\mathbf{X}^{(\ell)}, k)$;
4 **fin**
5 $\mathbf{H} \leftarrow$ la matrice d'association des $\mathbf{p}^{(\ell)}$ partitions ;
6 $\mathbf{H}^r \leftarrow$ La matrice d'association du modèle nul, calculée à partir des permutations aléatoires des partitions $\mathbf{p}^{(\ell)}$;
7 $\tau = \textit{moyenne}\,(\mathbf{H}^r)$;
8 **pour** $i, j = 1, \ldots, n$ **faire**
9 $\quad$ **si** $h_{ij} < \tau$ **alors**
10 $\quad\quad h_{ij} \leftarrow 0$;
11 $\quad$ **fin**
12 **fin**
13 $\mathbf{p}^* \leftarrow \mathcal{C}_2(\mathbf{H})$;
14 **retourner** $\mathbf{p}^*$;

contextes d'apprentissage automatique où elle contribue généralement à améliorer les résultats en combinant plusieurs modèles (Dietterich, 2000; Affeldt et al., 2020b,a). Elle permet une meilleure performance prédictive et un *clustering* plus robuste par rapport aux résultats obtenus avec un seul modèle (Strehl et Ghosh, 2002; Vega-Pons et Ruiz-Shulcloper, 2011; Berikov et Pestunov, 2017). En suivant le paradigme de la méthode ensemble, nous utilisons la matrice d'association $\mathbf{H}_{n\times n} = (h_{ij})$ pour calculer la partition consensuelle comme cela est décrit dans l'algorithme 1, où l'algorithme de *clustering* $\mathcal{C}_1$ est utilisé sur les matrices $\mathbf{X}_\ell$ pour obtenir les b partitions, tandis que $\mathcal{C}_2$ est utilisé sur la matrice $\mathbf{H}$ pour obtenir la partition consensuelle $\mathbf{p}^*$. h_{ij} désigne le nombre de partitions dans $\mathbf{p}^{(\ell)}, \ell = 1, \ldots, b$ qui affectent les individus i et j dans le même cluster. Afin d'exploiter au mieux la matrice $\mathbf{H}$, nous proposons d'utiliser une version simplifiée de l'approche proposée dans (Bassett et al., 2013). Notons que $\mathbf{H}$ peut être assimilé à une matrice d'adjacence de graphe. Pour partitionner la matrice $\mathbf{H}$, nous utilisons comme paramètre $\mathcal{C}_2$ un algorithme de *clustering* qui ne requiert pas nécessairement de définir le nombre de clusters à l'avance. Dans nos expériences, nous avons utilisé l'algorithme de Louvain (Blondel et al., 2008) pour obtenir la partition d'ensemble, qui a donné de meilleurs résultats que K-means appliqué sur la matrice $\mathbf{H}$. Dans l'étape 7 de l'algorithme 1, nous utilisons la $moyenne$ de la matrice $\mathbf{H}^r$ au lieu du max utilisé dans (Bassett et al., 2013). La raison en est que dans notre cas, le nombre de partitions (égal au nombre de couches b) est relativement petit.

Cela conduit la plus grande valeur de la matrice d'association aléatoire $\mathbf{H}^r$ à tendre facilement vers la plus grande valeur possible (i.e. le nombre de partitions b).

2.4 Étude expérimentale

Dans les expériences de *clustering*, nous utilisons 10 exécutions de K-means (MacQueen et al., 1967) (chacune avec n_init [1] fixé à 10) en tant que $\mathcal{C}_1$ dans l'algorithme 1 et l'algorithme de Louvain en tant que $\mathcal{C}_2$. Pour valider les résultats produits par le *clustering*, nous nous appuyons sur des mesures standard consacrées à l'évaluation de la qualité des partitions, à savoir l'information mutuelle normalisée (NMI) (Strehl et Ghosh, 2002) et l'indice de *Rand* ajusté (ARI) (Hubert et Arabie, 1985; Steinley, 2004).

2.4.1 Jeux de données et modèles utilisés

Les jeux de données utilisés pour les expériences de *clustering* sont décrits dans le tableau 1, où la donnée « Équilibre » est le rapport entre la taille de la plus petite et de la plus grande classe. Nous avons utilisé les jeux de données classic3 et classic4 de l'Université Cornell, le jeu de données de *news* de la BBC proposé dans (Greene et Cunningham, 2006) et un extrait aléatoire de DBPedia (Lehmann et al., 2015) et de AG-news [2] de taille 12 000 et 8 000 respectivement. À

TAB. 1 – Description des jeux de données.

	classic3	classic4	DBPedia	AG-news	BBC
Clusters	3	4	14	4	5
Équilibre	0.71	0.32	0.92	0.97	0.76
Documents	3 891	7 095	12 000	8 000	2 225

partir de chaque ensemble de documents, nous calculons plusieurs représentations contextuelles à partir de quatre modèles pré-entraînés qui sont les versions « base » et « large » de BERT (Devlin et al., 2019) et RoBERTa (Liu et al., 2019b) dont les caractéristiques sont données dans le tableau 2.

TAB. 2 – Descriptions des modèles Transformeurs.

	Nombre de couches	Nombre de dimensions	Taille du vocabulaire
BERT-base-cased	12	768	28 996
RoBERTa-base			50 265
BERT-large-cased	24	1 024	28 996
RoBERTa-large			50 265

2.4.2 Classification non supervisée couche par couche

Avant de procéder au partitionnement des représentations issues des Transformeurs, nous analysons les compressions en deux dimensions des plongements fournis par les différentes

1. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
2. http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

couches. Pour cela, nous utilisons t-SNE (Maaten et Hinton, 2008) avec les mêmes paramètres (perplexité à 15 et taux d'apprentissage à 200) à chaque exécution. Nous reportons l'évolution des couches dans la figure 2.

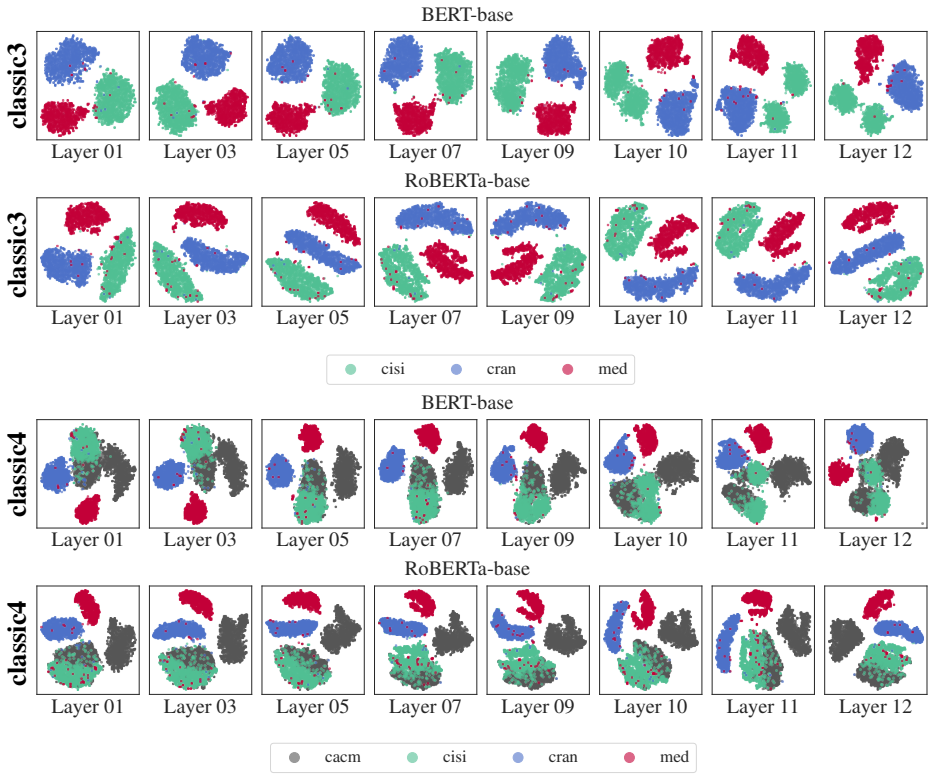

FIG. 2 – Projections à deux dimensions obtenues avec t-SNE. Les couleurs des points correspondent aux vrais labels dont nous disposons.

Nous pouvons observer une variabilité de la structure en classes selon les couches. Par exemple, pour BERT-base, la séparabilité des quatre classes de classic4, où les classes ne sont pas très bien séparées, se dégrade lorsqu'on se rapproche de la fin du réseau, avec notamment l'apparition d'une classe supplémentaire. Cependant, les représentations fournies par RoBERTa semblent permettre une meilleure séparabilité des classes de façon générale par rapport à BERT, avec notamment des résultats meilleurs au niveau des dernières couches.

Les résultats du *clustering* par couche sont présentés dans la figure 3 pour les cinq jeux de données. Celle-ci montre que réduire le nombre de dimensions à seulement 100 (ce qui constitue moins de 10% des caractéristiques des modèles « large ») conduit à une amélioration significative des performances, en particulier dans le cas de RoBERTa-base, pour lequel nous observons une augmentation d'au moins 0.54 sur la NMI pour toutes les couches de l'ensemble de données classic3. On peut également observer que, d'un jeu de données à l'autre, la meilleure couche n'est pas toujours la même. En effet, si nous prenons l'exemple de la base BERT, les meilleures couches pour les 5 jeux de données sont respectivement les couches 1, 11, 9, 2 et 1. De plus, on observe parfois plusieurs couches présentant de bons résultats, ce qui indique que

toutes les couches peuvent apporter des informations utiles, potentiellement différentes les unes des autres comme discuté dans (Peters et al., 2018b; Tenney et al., 2019; Vulić et al., 2020).

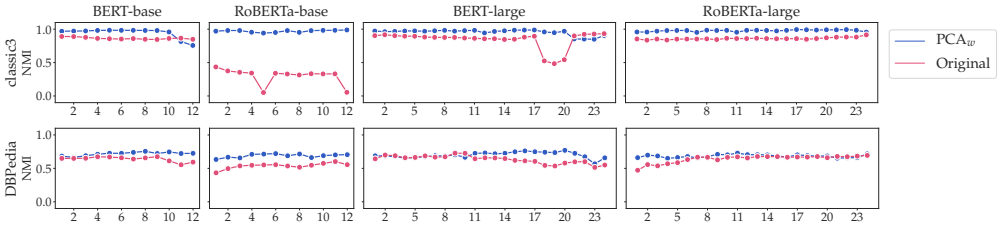

FIG. 3 – Performance du *clustering* (NMI) sur les représentations originales obtenues par chacune des couches des modèles pré-entraînés (en utilisant toutes les d dimensions des matrices $\mathbf{X}^{(\ell)}$) par rapport aux représentations réduites ($d' = 100$), avec $\ell = 1, \ldots, b$.

TAB. 3 – Valeurs de NMI du *clustering* de documents obtenu par les techniques de *clustering* multi-couche sur les quatre modèles Transformeur. La colonne « couche par couche » correspond à la moyenne des valeurs de NMI obtenues par chaque couche ℓ (en utilisant $\mathbf{X}^{(\ell)}$) et la valeur entre parenthèses au score obtenu par la meilleure couche, une couche qui ne peut malheureusement pas être identifiée en l'absence de labels (contexte non-supervisé).

Jeu de données	Modèle	BOW	Couche par couche		Av. dernière		4 dern. couches (orig.)			Toutes couches (orig.)			4 dern. couches (ACP$_w$)			Toutes couches (ACP$_w$)		
			Orig.	ACP$_w$	Orig.	ACP$_w$	AVG	Concat	ENS	AVG	Concat	ENS	AVG	Concat	ENS	AVG	Concat	ENS
classic3	BERT$_b$	0.95	0.86 (0.89)	0.94 (0.98)	0.87	0.82	0.85	0.87	0.87	0.85	0.87	0.88	0.95	0.78	**0.98**	**0.98**	0.92	**0.98**
	BERT$_b$		0.84 (0.93)	0.95 (0.99)	0.93	0.85	0.93	0.92	0.93	0.9	0.91	0.9	0.84	0.82	0.98	0.98	0.95	**0.99**
	RoBERTa$_b$		0.3 (0.43)	0.97 (0.99)	0.33	**0.98**	0.06	0.33	0.33	0.06	0.33	0.34	**0.98**	0.94	**0.98**	0.94	0.95	**0.98**
	RoBERTa$_l$		0.86 (0.91)	0.98 (0.99)	0.88	0.98	0.89	0.89	0.89	0.86	0.86	0.86	**0.99**	0.96	**0.99**	0.97	0.97	**0.99**
classic4	BERT$_b$	0.64	0.55 (0.64)	0.61 (0.68)	0.64	0.63	0.62	0.64	0.64	0.61	0.63	0.53	0.61	0.59	**0.74**	0.63	0.56	0.73
	BERT$_l$		0.51 (0.68)	0.59 (0.66)	0.4	0.61	0.52	0.54	0.58	0.68	0.53	0.53	0.56	0.57	0.64	0.66	0.6	**0.74**
	RoBERTa$_b$		0.24 (0.29)	0.55 (0.67)	0.24	0.61	0.24	0.24	0.24	0.23	0.24	0.25	0.59	0.65	0.7	0.58	0.55	**0.74**
	RoBERTa$_l$		0.52 (0.72)	0.6 (0.71)	0.54	0.63	0.55	0.55	0.54	0.51	0.52	0.54	0.67	0.61	**0.75**	0.69	0.67	**0.75**
DBPedia	BERT$_b$	0.69	0.64 (0.67)	0.72 (0.76)	0.55	0.72	0.65	0.61	0.57	0.69	0.67	0.69	**0.75**	0.74	0.71	0.74	0.72	**0.75**
	BERT$_l$		0.63 (0.73)	0.7 (0.77)	0.51	0.57	0.61	0.59	0.61	0.68	0.65	0.73	0.67	0.67	0.62	0.71	0.71	**0.76**
	RoBERTa$_b$		0.54 (0.6)	0.69 (0.72)	0.6	0.7	0.57	0.58	0.54	0.55	0.56	0.49	**0.73**	0.69	0.56	0.72	0.69	0.7
	RoBERTa$_l$		0.64 (0.69)	0.68 (0.73)	0.68	0.66	0.69	0.68	0.68	0.69	0.66	0.68	0.67	0.7	0.58	0.66	0.7	**0.73**
AG-news	BERT$_b$	0.46	0.39 (0.48)	0.43 (0.46)	0.33	0.39	0.4	0.37	0.39	0.44	0.42	0.41	0.43	0.44	0.36	0.43	0.36	**0.54**
	BERT$_l$		0.3 (0.57)	0.38 (0.53)	0.0	0.06	0.17	0.01	0.0	0.49	0.21	0.49	0.11	0.03	0.0	0.5	0.2	**0.54**
	RoBERTa$_b$		0.39 (0.44)	0.47 (0.5)	0.44	0.42	0.43	0.44	0.43	0.41	0.42	0.41	0.46	0.47	0.47	0.48	0.44	**0.58**
	RoBERTa$_l$		0.44 (0.53)	0.46 (0.54)	0.52	0.49	0.53	0.53	0.52	0.51	0.49	0.46	0.46	0.46	0.53	0.53	0.47	**0.59**
BBC	BERT$_b$	0.75	0.55 (0.77)	0.61 (0.67)	0.47	0.59	0.46	0.45	0.45	0.6	0.51	0.55	0.63	0.66	0.74	0.54	0.61	**0.8**
	BERT$_l$		0.67 (0.85)	0.57 (0.66)	0.78	0.45	0.82	0.82	0.79	0.79	0.78	0.79	0.43	0.4	0.62	0.53	0.5	**0.88**
	RoBERTa$_b$		0.36 (0.52)	0.56 (0.6)	0.38	0.5	0.37	0.38	0.39	0.34	0.38	0.38	0.53	0.54	0.64	0.62	0.48	**0.83**
	RoBERTa$_l$		0.66 (0.87)	0.5 (0.58)	**0.86**	0.44	**0.86**	**0.86**	**0.86**	0.74	0.74	0.74	0.52	0.5	0.65	0.51	0.57	0.79
Rang moyen		5.58	-	-	10.35	9.05	9.55	9.35	9.75	9.68	10.45	9.88	7.32	8.32	6.10	5.58	7.45	**1.60**

2.4.3 Classification non supervisée multi-couche

Étant donné que les résultats obtenus peuvent varier considérablement d'une couche à l'autre, comme le montre clairement la figure 3, et que déterminer laquelle est la meilleure est très difficile en l'absence de labels lorsqu'il s'agit de jeux de données réels, nous proposons d'utiliser simultanément toutes les matrices de données $\mathbf{X}^{(\ell)}$ $\ell = 1, \ldots, b$ fournies par le réseau comme décrit dans les sections 2.2 et 2.3. Le tableau 3 présente la NMI obtenue par chaque technique

en supposant que le nombre de clusters k est connu. Nous comparons l'approche multi-couche à plusieurs approches antérieures. La première est l'utilisation d'une représentation en sac de mots (BOW) en entrée d'un algorithme Spherical K-means (Buchta et al., 2012), connu pour être plus adapté aux données directionnelles par rapport à K-means. Deux autres approches de base sont l'utilisation de l'avant-dernière couche (Xiao, 2018; Devlin et al., 2019) ainsi que la combinaison des quatre dernières couches (Devlin et al., 2019). Le **rang moyen** correspond à la moyenne de tous les rangs attribués à une méthode donnée en fonction de ses performances par rapport aux autres méthodes, sur toutes les combinaisons modèle-données. Nous observons tout d'abord l'efficacité de ACP_w lorsque l'on compare les scores obtenus avec les vecteurs originaux et leur version réduite (par exemple pour AVG, Concat et ENS en utilisant toutes les couches, on passe d'un **rang moyen** de 9.68, 10.45 et 9.88 à 5.58, 7.45 et 1.6 respectivement). Les résultats obtenus montrent également que l'utilisation de l'avant-dernière couche n'est pas fiable, car son efficacité dépend fortement du modèle et du jeu de données. La combinaison des quatre dernières couches est plus efficace mais présente des performances inférieures à l'utilisation de l'ensemble des couches. Cela suggère que les informations utiles fournies par les plongements contextuels sont différentes d'une couche à l'autre. De plus, toute l'information utile semble être efficacement capturée par les dimensions réduites (voir les résultats Concat, où l'on passe de $d \in \{9\ 216, 24\ 576\}$ dimensions à seulement 100). De plus, nos résultats montrent clairement un avantage significatif de la technique d'ensemble sur les autres approches de combinaison de couches, présentant les résultats les plus élevés en termes de NMI et dont le **rang moyen** est de loin le meilleur.

3 Partitionnement avec un nombre de classes estimé

Tous les résultats présentés précédemment reposent sur l'hypothèse que le nombre exact de classes est connu à l'avance, ce qui n'est pas toujours vrai dans la réalité. Par ailleurs, un autre avantage significatif offert par l'approche proposée est l'utilisation d'un algorithme dont le nombre de classes n'est pas connu *a priori* (Louvain dans nos expériences). Cela signifie que le consensus renvoie une partition avec des classes qui respectent autant que possible les partitionnements d'origine, sans nécessairement fournir le même nombre de classes que les partitions d'entrée. C'est donc une bonne alternative lorsque le nombre exact de classes est inconnu. De plus, le nombre de classes pour chaque partition n'est pas nécessairement le même, ce qui est une caractéristique intéressante du *clustering ensemble*. Afin de bénéficier de cette propriété, étant donné un ensemble $\mathcal{K}$ de certaines valeurs sélectionnées de k, on s'assure que chaque exécution de K-means prend en entrée une valeur $k \in \mathcal{K}$ tout en couvrant autant que possible l'ensemble des valeurs. Nous utilisons dans nos expériences $\mathcal{K} = [k_r - 5, k_r + 5]$ où k_r est le nombre réel de classes, e.g. pour le jeu de données DBPedia où $k_r = 14$ en utilisant un modèle « base » à 12 couches, la liste des 12 valeurs de k pourrait éventuellement être $\{9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 9\}$, obtenant 12 partitions $\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(\ell)}, \ldots, \mathbf{p}^{(12)}$ à partir desquelles nous calculons la partition consensuelle $\mathbf{p}^*$ comme décrit dans l'algorithme 1. La partition $\mathbf{p}^*$ regroupe alors dans une même classe les individus qui sont habituellement regroupés dans les partitions d'entrée sans avoir la contrainte d'un k fixe. Cela garantit une performance de *clustering* robuste tout en estimant automatiquement le nombre de classes. Le tableau 4 montre les résultats de l'algorithme d'ensemble (en utilisant les représentations réduites) obtenus avec des valeurs variables de $k \in [k_r - 5, k_r + 5]$ (k_r est donné dans le tableau

1). Nous observons à partir du tableau 4 que les performances ne sont pas significativement

TAB. 4 – Performances obtenues par le *clustering ensemble* en utilisant toutes les couches avec un nombre estimé de classes.

Modèle	classic3			classic4			DBPedia			AG-news			BBC		
	$\hat{k}$	NMI	ARI	$\hat{k}$	NMI	ARI	$\hat{k}$	NMI	ARI	$\hat{k}$	NMI	ARI	$\hat{k}$	NMI	ARI
BERT-base	3	0.98	0.99	4	0.74	0.52	8	0.71	0.49	4	0.55	0.56	4	0.69	0.62
BERT-large	3	0.98	0.99	4	0.73	0.52	9	0.77	0.54	4	0.5	0.45	4	0.74	0.64
RoBERTa-base	3	0.98	0.99	5	0.79	0.65	10	0.78	0.59	4	0.52	0.49	4	0.74	0.65
RoBERTa-large	3	0.98	0.99	4	0.74	0.53	7	0.68	0.41	4	0.59	0.51	4	0.75	0.65

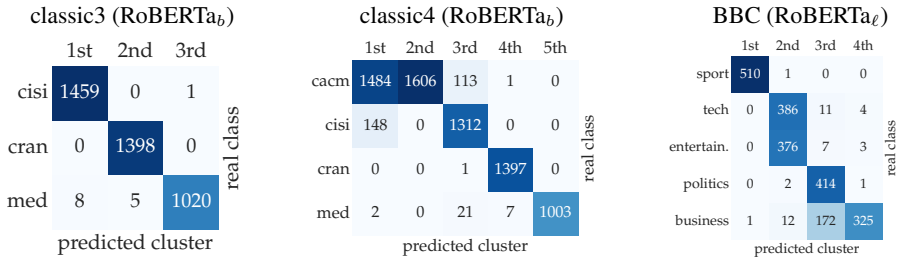

FIG. 4 – Matrices de confusion obtenues par le *classeing ensemble* avec un nombre estimé de classes (comme décrit dans la section 3).

altérées, même lorsque le nombre estimé de classes n'est pas égal à k_r. Pour classic3, classic4 et AG-news, le *clustering ensemble* avec des valeurs de k variables trouve un nombre de classes toujours égal à k_r, sauf pour classic4 en utilisant RoBERTa-base, où une partition de 5 classes est trouvée mais avec une NMI et ARI élevées, ce qui est expliqué par le fait que la classe supplémentaire correspond à la séparation de la classe « cacm » (cf. figure 4). A l'inverse, le nombre de classes est sous-estimé pour le jeu de données BBC (4 clusters au lieu de 5), où les classes « tech » et « entertainement » semblent avoir été fusionnées. Pour DBPedia, les représentations fournies par RoBERTa-base sont celles présentant une estimation de k la plus proche de k_r, ainsi qu'un score NMI élevé. Dans ce cas, comme pour BBC, certaines classes sont fusionnées telles que « animal » avec « plant » et « film » avec « written work ».

4 Conclusion

Dans cet article, nous avons étudié les performances des représentations obtenues à partir de quatre modèles Transformeur pré-entraînés lorsqu'ils sont utilisés comme entrée dans un algorithme de *clustering* de documents. Ceci est une contribution à l'utilisation des plongements Transformeur dans le cadre de l'apprentissage non-supervisé.

Nos expérimentations montrent que la méthode de *clustering ensemble* proposée combinée à une réduction basée sur l'ACP permet de tirer le meilleur parti des modèles Transformeur, en obtenant des performances encore meilleures que celles fournies par la meilleure couche qui, rappelons le, n'est pas identifiable dans un contexte non-supervisé. Les pistes de recherches futures incluent la poursuite de l'amélioration de la procédure *ensemble* proposée, en particulier

l'estimation du nombre de classes et l'expérimentation d'autres techniques de réduction de dimension. Une autre perspective consiste à évaluer l'impact du ré-entraînement (*fine-tuning*) des modèles Transformeur sur le *clustering* de texte.

Références

Affeldt, S., L. Labiod, et M. Nadif (2020a). Ensemble block co-clustering : a unified framework for text data. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 5–14.

Affeldt, S., L. Labiod, et M. Nadif (2020b). Spectral clustering via ensemble deep autoencoder learning (sc-edae). *Pattern Recognition 108*, 107522.

Ait-Saada, M., F. Role, et M. Nadif (2021a). How to leverage a multi-layered transformer language model for text clustering : an ensemble approach. In *CIKM*, pp. 2837–2841.

Ait-Saada, M., F. Role, et M. Nadif (2021b). Unsupervised methods for the study of transformer embeddings. In *Advances in Intelligent Data Analysis XIX*, Cham, pp. 287–300. Springer International Publishing.

Bassett, D. S., M. A. Porter, N. F. Wymbs, S. T. Grafton, J. M. Carlson, et P. J. Mucha (2013). Robust detection of dynamic community structure in networks. *Chaos : An Interdisciplinary Journal of Nonlinear Science 23*(1), 013142.

Berikov, V. et I. Pestunov (2017). Ensemble clustering based on weighted co-association matrices : Error bound and convergence properties. *Pattern Recognition 63*, 427–436.

Blondel, V. D., J.-L. Guillaume, R. Lambiotte, et E. Lefebvre (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics : theory and experiment 2008*(10), P10008.

Buchta, C., M. Kober, I. Feinerer, et K. Hornik (2012). Spherical k-means clustering. *Journal of statistical software 50*(10), 1–22.

Clark, K., U. Khandelwal, O. Levy, et C. D. Manning (2019). What does BERT look at ? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP*, pp. 276–286.

Devlin, J., M.-W. Chang, K. Lee, et K. Toutanova (2019). BERT : Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, pp. 4171–4186. Association for Computational Linguistics.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer.

Greene, D. et P. Cunningham (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pp. 377–384. ACM Press.

Hao, Y., L. Dong, F. Wei, et K. Xu (2019). Visualizing and Understanding the Effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*

(EMNLP-IJCNLP), Hong Kong, China, pp. 4141–4150. Association for Computational Linguistics.

Hubert, L. et P. Arabie (1985). Comparing partitions. *Journal of classification 2*(1), 193–218.

Kovaleva, O., A. Romanov, A. Rogers, et A. Rumshisky (2019). Revealing the Dark Secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, pp. 4364–4373. Association for Computational Linguistics.

Lehmann, J., R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al. (2015). Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web 6*(2), 167–195.

Li, X. L. et J. Eisner (2019). Specializing word embeddings (for parsing) by information bottleneck. *arXiv preprint arXiv:1910.00163*.

Liu, N. F., M. Gardner, Y. Belinkov, M. E. Peters, et N. A. Smith (2019a). Linguistic Knowledge and Transferability of Contextual Representations. In *Proceedings of the 2019 Conference of the North*, Minneapolis, Minnesota, pp. 1073–1094. Association for Computational Linguistics.

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, et V. Stoyanov (2019b). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Maaten, L. v. d. et G. Hinton (2008). Visualizing data using t-sne. *Journal of machine learning research 9*(Nov), 2579–2605.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Volume 1, pp. 281–297. Oakland, CA, USA.

Mu, J. et P. Viswanath (2018). All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*.

Peters, M., M. Neumann, L. Zettlemoyer, et W.-t. Yih (2018b). Dissecting Contextual Word Embeddings: Architecture and Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, pp. 1499–1509. Association for Computational Linguistics.

Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, et L. Zettlemoyer (2018a). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana, pp. 2227–2237. Association for Computational Linguistics.

Raunak, V., V. Gupta, et F. Metze (2019). Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pp. 235–243.

Reimers, N. et I. Gurevych (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language*

Processing (EMNLP-IJCNLP), Hong Kong, China, pp. 3980–3990. Association for Computational Linguistics.

Steinley, D. (2004). Properties of the hubert-arable adjusted rand index. *Psychological methods 9*(3), 386.

Strehl, A. et J. Ghosh (2002). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research 3*(Dec), 583–617.

Tenney, I., D. Das, et E. Pavlick (2019). BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, pp. 4593–4601. Association for Computational Linguistics.

van Aken, B., B. Winter, A. Löser, et F. A. Gers (2019). How Does BERT Answer Questions?: A Layer-Wise Analysis of Transformer Representations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, Beijing China, pp. 1823–1832. ACM.

Vega-Pons, S. et J. Ruiz-Shulcloper (2011). A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence 25*(03), 337–372.

Vulić, I., E. M. Ponti, R. Litschko, G. Glavaš, et A. Korhonen (2020). Probing Pretrained Language Models for Lexical Semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, pp. 7222–7240. Association for Computational Linguistics.

Xiao, H. (2018). bert-as-service. `https://github.com/hanxiao/bert-as-service`.

Xu, W., X. Liu, et Y. Gong (2003). Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 267–273.

Zhang, T., V. Kishore, F. Wu, K. Q. Weinberger, et Y. Artzi (2020). Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Summary

Pre-trained Transformer-based word embeddings are now widely used in text mining where they are known to significantly improve supervised tasks such as text classification, named entity recognition and question answering. Since the Transformer models create several different embeddings for the same input, one at each layer of their architecture, various studies have already tried to identify those of these embeddings that most contribute to the success of the above-mentioned tasks. In contrast the same performance analysis has not yet been carried out in the unsupervised setting. In this paper we evaluate the effectiveness of Transformer models on the important task of text clustering. In particular, we present a clustering ensemble approach that harnesses all the network's layers. Numerical experiments carried out on real datasets with different Transformer models show the effectiveness of the proposed method compared to several baselines.