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 modèles *Transformeurs* créent des plongements différents pour la même entrée, un à chaque couche de leur architecture. Diverses études ont déjà tenté d'identifier les plongements qui contribuent le plus au succès des tâches de classification supervisée. 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 *Transformeurs* sur l'importante tâche de classification non supervisée de documents. Nous présentons une approche *clustering ensemble* qui exploite toutes les couches du réseau. Des expériences menées sur des ensembles de données réels avec différents modèles montrent l'efficacité de la méthode proposée par rapport à plusieurs stratégies habituellement utilisées. Cet article est une restitution du papier (Ait-Saada et al., 2021).

1 Introduction

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 (Tenney et al., 2019), en utilisant des plongements issus de modèles Transformeur pré-entraînés en entrée d'une suite de tâches de traitement automatique de la langue, les auteurs ont observé 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). 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

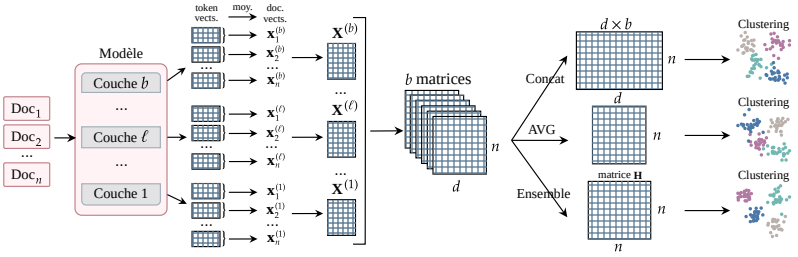

FIG. 1 – Description des différentes manières de combiner les couches ℓ d'un modèle Transformeur. $\mathbf{x}_i^{(\ell)}$ est la représentation du document calculé en faisant la moyenne des représentations (obtenues à la couche ℓ) des tokens contenus dans le document i. Ce vecteur forme la i-ième ligne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

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.

Ainsi, 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. 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 ont montré qu'il était possible de réduire de moitié les dimensions sans altérer de manière significative les performances. 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 (Xiao, 2018), 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 $n \times d$ (Figure 1).

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 $\mathbf{X}^{(\ell)}$, $\ell = 1, \ldots b$ 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. 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 $d$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 $b \times d$ 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 avant d'exécuter un algorithme de *clustering*, obtenant des représentations de taille $d' = 100$.

1.1 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 14). L'approche *ensemble* a été considérée dans différents contextes d'apprentissage (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). 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 14, 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 $\mathbf{p}^{(\ell)}, \ell = 1, \dots, b$ 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 14, 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).

1.2 Étude expérimentale

Dans les expériences de *clustering*, nous utilisons 10 exécutions de K-means (chacune avec n_init [1] fixé à 10) en tant que $\mathcal{C}_1$ dans l'algorithme 14 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,

1. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

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 $\mathbf{p}^*$
pour $\ell = 1, \ldots, b$ **faire**
  $\mathbf{X}^{(\ell)} \leftarrow$ plongements de documents calculés avec $\mathcal{M}(\mathcal{D})$ à chaque couche ℓ (Figure 1) ;
  $\mathbf{p}^{(\ell)} \leftarrow \mathcal{C}_1(\mathbf{X}^{(\ell)}, k)$;
fin
$\mathbf{H} \leftarrow$ la matrice d'association des $\mathbf{p}^{(\ell)}$ partitions ;
$\mathbf{H}^r \leftarrow$ La matrice d'association du modèle nul, calculée à partir des permutations aléatoires des partitions $\mathbf{p}^{(\ell)}$;
$\tau = \textit{moyenne}\,(\mathbf{H}^r)$;
pour $i, j = 1, \ldots, n$ **faire**
  si $h_{ij} < \tau$ **alors**
    $h_{ij} \leftarrow 0$;
  fin
fin
$\mathbf{p}^* \leftarrow \mathcal{C}_2(\mathbf{H})$;
retourner $\mathbf{p}^*$;

1985). 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, 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 de base (12 couches) et grande (24 couches) de BERT Devlin et al. (2019) et RoBERTa Liu et al. (2019b), avec une taille de vocabulaire de 28 996 pour BERT et 50 265 pour RoBERTa.

Les résultats du *clustering* par couche sont présentés dans la figure 2.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 BERT-base, les meilleures couches pour les 5 jeux de données sont respectivement les couches 1, 11, 9, 2 et 1. De plus, on observe parfois

2. http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

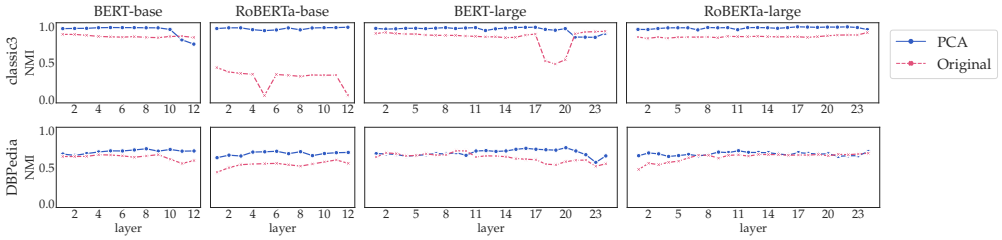

FIG. 2 – Performance du *clustering* (NMI) sur les représentations originales obtenues par chacune des couches des modèles pré-entraînés (en utilisant toutes les d dimensions des matrices $\mathbf{X}^{(\ell)}$) par rapport aux représentations réduites ($d' = 100$), avec $\ell = 1, \ldots, b$.

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 (Tenney et al., 2019). Étant donné que les résultats obtenus peuvent varier considérablement d'une couche à l'autre, comme le montre clairement la figure 2, 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 1 et 1.1.

Le tableau 2 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

TAB. 2 – 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)			Toutes couches (ACP)		
			Orig.	ACP	Orig.	ACP	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	**0.80**
	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**

de l'avant-dernière couche (Xiao, 2018) 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 la réduction de dimension basée sur l'ACP 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.

2 Partitionnement avec un nombre de classes estimé

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*. 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 14. 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 3 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 3 que les performances ne sont pas significativement 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 3). A l'inverse, le nombre de classes est sous-estimé pour le jeu

TAB. 3 – 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

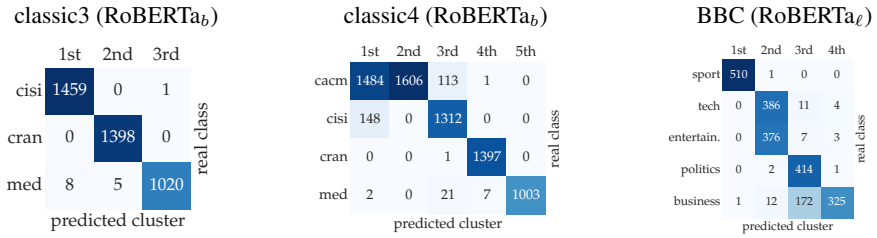

FIG. 3 – Matrices de confusion obtenues par le *classeing ensemble* avec un nombre estimé de classes (comme décrit dans la section 2).

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 ».

3 Conclusion

Notre proposition et 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.

Remerciements. Ce travail a été soutenu par une subvention de projet émergence 2021 IDEX-Université de Paris (Spectrans) et l'Agence Nationale de la Recherche (ANR-19-CE23-0002).

Références

Affeldt, S., L. Labiod, et M. Nadif (2020a). Ensemble block co-clustering : a unified framework for text data. In *CIKM*, 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 (2021). How to leverage a multi-layered transformer language model for text clustering : an ensemble approach. In *CIKM*, pp. 2837–2841.

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.

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).

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

Devlin, J., M.-W. Chang, K. Lee, et K. Toutanova (2019). BERT : Pre-training of deep bidirectional transformers for language understanding. In *ACL*, pp. 4171–4186.

Greene, D. et P. Cunningham (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering. In *ICML*, pp. 377–384.

Hao, Y., L. Dong, F. Wei, et K. Xu (2019). Visualizing and Understanding the Effectiveness of BERT. In *EMNLP-IJCNLP*, pp. 4141–4150.

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

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.

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

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*.

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

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 *ACL*, Florence, Italy, pp. 4593–4601.

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 *CIKM*, pp. 1823–1832.

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

Summary

In this paper we evaluate the effectiveness of Transformer models on the important task of text clustering. 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.