How to Leverage a Multi-layered Transformer Language Model for Text Clustering: an Ensemble Approach

Mira Ait-Saada, François Role, Mohamed Nadif

How to Leverage a Multi-layered Transformer Language Model for Text Clustering: an Ensemble Approach

Mira Ait-Saada
mira.ait-saada@u-paris.fr
Université de Paris, Centre Borelli
& Groupe Caisse des Dépôts
Paris, France

François Role
francois.role@u-paris.fr
Université de Paris
CNRS, Centre Borelli UMR 9010
Paris, France

Mohamed Nadif
mohamed.nadif@u-paris.fr
Université de Paris
CNRS, Centre Borelli UMR 9010
Paris, France

ABSTRACT

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 and 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.

CCS CONCEPTS

• **Computing methodologies** → **Unsupervised learning**; **Natural language processing**.

KEYWORDS

Transformer-based Language Models, Unsupervised Learning, Document Embeddings, Clustering Ensemble, Dimension Reduction

ACM Reference Format:
Mira Ait-Saada, François Role, and Mohamed Nadif. 2021. How to Leverage a Multi-layered Transformer Language Model for Text Clustering: an Ensemble Approach. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3482121

1 INTRODUCTION AND RELATED WORK

Starting with BERT [7], Transformer-based contextualized word embeddings provided by neural language models have been increasingly used as the initial input to many NLP applications where they greatly contribute to achieving impressing performance levels.

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482121

A Transformer model produces several representations for each word (one at each layer of the network architecture) and studies in the realm of supervised learning have tried to determine the kind of information captured by these different layers. For example, using pre-trained Transformer embeddings as input to a suite of NLP tasks, the authors in [18, 23] have agreed that early layers encode most local syntactic phenomena while more complex semantics appear at the higher layers. On the other hand, looking more specifically at the generalization capabilities of contextualized word representations (including ELMo, BERT and OpenAI Transformer algorithms), Liu et al. [14] have observed that Transformers' middle layers present a better transferability while Hao et al. [10] observed that the early layers of BERT-large (24 layers) are more invariant across tasks than the higher layer and hence more transferable. In another line of research, some studies have concentrated on the impact of fine-tuning and have experimentally verified that the closer we get to the last layer, the more task specific the representations are [12, 24].

The main takeaway of these studies is that embeddings available at the different layers clearly capture different information, thus leading to very different results when used as input to a given text mining task. The problem is that it is not possible to know in advance which one will help to best perform on a given task. When using pre-trained embeddings, a common empirical rule is to exclude the last layer on the assumption that it is biased to the training targets. The very first layers are also commonly excluded since they are deemed to be too close to the original word information. Another approach for selecting a layer to perform a given task is to utilize a labeled dataset as a development set to determine the best layer to use for new datasets [29]. We show that this approach is not optimal in our case, since the best layer is often different from one dataset to another. In addition, we believe that semantic features can be very helpful in text clustering, and it has been observed in [23] that, unlike syntactic information, which is generally concentrated in a few layers, semantic features are spread out across the entire network. This is why, instead of choosing a unique layer, we prefer leveraging all the representations provided by Transformer models to perform unsupervised learning. To achieve that, we propose to separately cluster the document representations computed at each layer, then deduce a consensual partition, taking advantage of all the information provided at each level of the deep network. In order to evaluate our approach, we compare it to formerly used baselines including the concatenation and averaging of layers, the use of the second-to-last layer as well as the combination of the four last layers. We also compare our results to those obtained with a standard Bag-Of-Words representation.

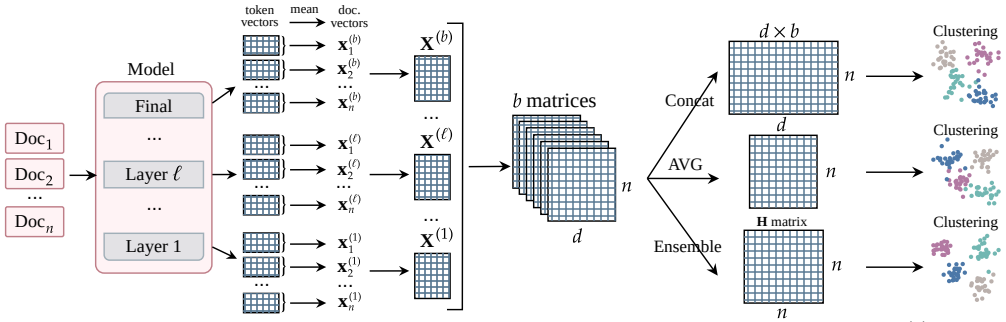

Figure 1: Different ways of combining the ℓ layers' embeddings of a Transformer language model. $\mathbf{x}_i^{(\ell)}$ is the document vector computed by averaging the representations (obtained at layer ℓ) of the tokens contained in document i. This vector forms the i-th row of the $\mathbf{X}^{(\ell)}$ matrix, which is the representation of the dataset at layer ℓ.

Also, we investigate the effect of dimension reduction on the Transformer-based embeddings, a topic which has rarely been studied so far. For a review of this question in the context of word embeddings that predate BERT such as Word2vec, fastText and GloVe, the reader is referred to [19] who showed that it was possible to half vector dimensions without significantly altering performance. The present paper aims at contributing to fill this gap in the context of Transformer embeddings. We show in Section 2 that combining clustering ensemble and dimension reduction allows to significantly increase clustering performance on several real datasets. Section 3 then highlights an important advantage of the clustering ensemble, namely the effective estimation of the number of clusters along with an efficient partitioning of data samples, which is very useful when the exact number of clusters is not known.

2 CLUSTERING TRANSFORMER EMBEDDINGS

For a dataset of n documents and a Transformer model with b layers, one obtains b dense representations of size d, one from each layer. Given a layer ℓ, the representation of the i-th document of the dataset is obtained from the embeddings of its 512 first tokens that are pooled together using the mean function (as suggested in [20, 27]), hence obtaining a vector $\mathbf{x}_i^{(\ell)}$ of size d, which constitutes the i-th row of the matrix $\mathbf{X}^{(\ell)}$. The dataset can then be represented by b matrices $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(b)}$ of size $n \times d$ as shown in Figure 1. We call a partition the result provided by one of the clustering runs and it contains n labels where the i-th label corresponds to the cluster to which the i-th document is assigned. For a model with b layers, one can think of running the clustering algorithm on each of the $\mathbf{X}^{(\ell)}$ matrices, and pick the one that yields the "best" result. However, in the unsupervised setting where no labels are available, there is no easy way of knowing which matrix $\mathbf{X}^{(\ell)}$ has given the best result. Hence, we propose and describe two main ways of leveraging the various representations provided by a Transformer model: (1) by aggregating the $\mathbf{X}^{(\ell)}$, $\ell = 1, \ldots b$ matrices; (2) by using the $\mathbf{X}^{(\ell)}$ matrices individually as part of an ensemble approach. We also assess the performance obtained when using a PCA-based dimension reduction step, reducing the dimensions to $d' = 100$.

2.1 Post-processing

The use of linear dimension reduction is commonplace in NLP and has shown promising improvements on various representations [17, 19, 28]. In our study, we investigate the power of principal components in reducing the dimension of dense text representations while preserving the information they contain. In particular, we observed that the use of the first principal components does not alter the performance, even when compressing the original vectors to 10% of the dimensions. More importantly, we show that an additional whitening operation applied on the principal components leads to surprising improvements of the clustering performance while drastically reducing the dimensionality. As an alternative to removing the dominant principal components (PCs) [17, 19], the whitening operation allows to normalize the PCs to unit variance, thus reducing the impact of the first components and producing vectors of better quality. It consists in building a reduced representation $\mathbf{Y}$ whereby each value is computed as $y_{ij} = \mathbf{x}_i \mathbf{w}_j^\top / \sqrt{\sigma_j}$, where $\mathbf{w}_j$ is the jth eigen vector of $\mathbf{X}^\top \mathbf{X}$ and σ_j its jth eigen value. In the remainder of this paper, PCA refers to the use of the d' first whitened principal components.

2.2 Aggregating the Layered Representations

For a document i, the first combination method consists in averaging the b vectors $\mathbf{x}_i^{(\ell)}$, $\ell = 1, \ldots b$, thus obtaining a unique vector of size d, as experienced in [26]. We refer to this method as `AVG`. The second method, which we call `Concat`, consists in concatenating the b vectors $\mathbf{x}_i^{(\ell)}$, which results in a unique vector of size $b \times d$, as performed in [7] using the last layers. On top of these aggregated representations, we optionally perform a PCA-based dimension reduction before running the clustering algorithm, obtaining representations of size $d' = 100$.

2.3 Using a Clustering Ensemble Approach

Another way of combining the information provided by all layers, is not to physically aggregate them, but to rely on a consensus procedure. This clustering ensemble or consensus is referred to as `ENS`. Ensemble learning has been considered in different machine learning contexts where it generally helps in improving results by combining several models [1, 2, 8]. The ensemble approach allows a better predictive performance and a more robust clustering as compared to the results obtained with a single model [4, 22, 25]. Following the ensemble paradigm, we use the association matrix $\mathbf{H}_{n \times n} = (h_{ij})$ to compute the consensus partition as described in Algorithm 1, where the clustering algorithm C_1 is used on the $\mathbf{X}_\ell$ matrices to obtain the b partitions, while C_2 is used on $\mathbf{H}$ to

Algorithm 1: Clustering Ensemble

input : A dataset $\mathcal{D}$; a Transformer model $\mathcal{M}$ of b layers, two clustering algorithms C_1 and C_2; the number of clusters k

output : A consensual clustering partition $\mathbf{p}^*$

1 **foreach** $\ell = 1, \ldots, b$ **do**
2 $\quad \mathbf{X}^{(\ell)} \leftarrow$ document embeddings computed using $\mathcal{M}(\mathcal{D})$ at the ℓ-th layer (as shown in Figure 1);
3 $\quad \mathbf{p}^{(\ell)} \leftarrow C_1(\mathbf{X}^{(\ell)}, k)$;
4 **end**
5 $\mathbf{H} \leftarrow$ the association matrix of the $\mathbf{p}^{(\ell)}$ partitions;
6 $\mathbf{H}^r \leftarrow$ The null-model association matrix from random permutations of the partitions $\mathbf{p}^{(\ell)}$;
7 $\tau = average(\mathbf{H}^r)$;
8 **foreach** *i, j from* 1 *to n* **do**
9 $\quad$ **if** $h_{ij} < \tau$ **then**
10 $\quad\quad h_{ij} \leftarrow 0$;
11 $\quad$ **end**
12 **end**
13 $\mathbf{p}^* \leftarrow C_2(\mathbf{H})$;
14 **return** $\mathbf{p}^*$;

compute the consensus partition $\mathbf{p}^*$. h_{ij} denotes the number of partitions within $\mathbf{p}^{(\ell)}, \ell = 1, ..., b$ that assign the individuals i and j to the same cluster. In order to leverage $\mathbf{H}$, we propose to use a simplified version of the approach proposed in [3]. Note that $\mathbf{H}$ can be assimilated to a graph adjacency matrix. To cluster the $\mathbf{H}$ matrix, we use as the parameter C_2 a clustering algorithm that doesn't necessarily require to set the number of clusters in advance. In our experiments we used the Louvain algorithm [5] to obtain the consensus partition, which worked better than K-means on $\mathbf{H}$. In step 7 of Algorithm 1, we use the *mean* of the *null association* matrix $\mathbf{H}^r$ instead of the *max* used in [3]. The reason is that in our case, the number of partitions (equal to the number of layers b) is relatively small. This conducts the largest value of the randomly shuffled association matrix to tend easily to the largest possible value (i.e. the number of partitions b).

2.4 Experimental Study and Compared Results

In the clustering experiments, we used 10 runs of K-means [16] (each one with `n_init`[1] set to 10) as C_1 in Algorithm 1 and the Louvain Algorithm as C_2. To validate the results produced by the clustering we rely on standard external measures devoted to assessing cluster quality namely Normalized Mutual Information (NMI) [22] and Adjusted Rand Index (ARI) [11, 21].

2.4.1 Datasets and Models Used. The datasets used for clustering experiments are described in Table 1, where the balance is the ratio between the smallest and largest cluster sizes. We used classic3 and classic4 datasets of Cornell University, the BBC news dataset proposed in [9] and random extracts of DBPedia [13] and AG-news[2] of size 12,000 and 8,000 respectively. From each set of documents, we compute multiple contextual representations from 4 pre-trained models which are the base (12 layers) and large (24 layers) versions of BERT [7] and RoBERTa [15], with a vocabulary size of 28,996 for BERT and 50,265 for RoBERTa.

Table 1: Datasets' description.

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

2.4.2 Layer-wise Clustering Results. Layer-wise clustering results are presented in Figure 2 for two datasets. The Figure shows that reducing the number of dimensions to only 100 (which constitutes less than 10% of the large model's features) leads to a significant improvement of performance, especially in the case of RoBERTa-base, for which we observe an increase of at least 0.54 in NMI score for all layers on the classic3 dataset. We can also observe that, from a dataset to another, the best layer is not always the same. Indeed, if we take the example of BERT-base, the best layers for the 5 datasets are respectively the 1-st, 11-th, 9-th, 2-nd and 1-st. Moreover, we sometimes observe several layers presenting good results, which indicates that all layers can bring useful information, potentially different from one to another as discussed in [18, 23, 26].

2.4.3 Multi-layer Clustering Results. Since the obtained results may greatly vary with each layer, as clearly visible in Figure 2, and since determining which one is the best is very difficult in the absence of labels, we propose to simultaneously use all the $\mathbf{X}^{(\ell)}$ data matrices $\ell = 1, \ldots, b$ provided by the network as describes in sections 2.2 and 2.3. Table 2 presents the NMI obtained by each technique assuming that the number of clusters k is known. We compare the all-layered approach to several baselines. The first one is the use of a Bag-Of-Words (`BOW`) representation as input to a Spherical K-means [6] algorithm, known to be well suited to directional sparse data compared to K-means. Two other baselines are the use of the second-to-last layer [7, 27] as well as the combination of the four last layers [7]. The *mean rank* corresponds to the average of all the ranks assigned to a given method based on its performance compared to the other methods, over all the model-dataset combinations. A first observation is the effectiveness of the PCA-based dimension reduction when comparing the scores obtained with raw vectors and their reduced version (e.g. for `AVG`, `Concat` and `ENS` using all layers, we move from a *mean rank* of 9.68, 10.45 et 9.88 to 5.58, 7.45 et 1.6 resp.). The results also show that the use of the second-to-last layer is not reliable, as its effectiveness highly depends on the model and the dataset. Combining the four last layers is more effective but presents lower performance than the use of the all of the layers. This suggests that the useful information provided by the embeddings is different from one layer to another. Also, all of the useful information seems to be efficiently captured by the reduced dimensions (see the `Concat` results, for which we move from $d \in \{9\,216, 24\,576\}$ to only 100). Moreover, our results clearly show a significant advantage of the ensemble technique over the other approaches of combining layers, presenting the highest results in terms of NMI and the lowest *mean rank* by far.

[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

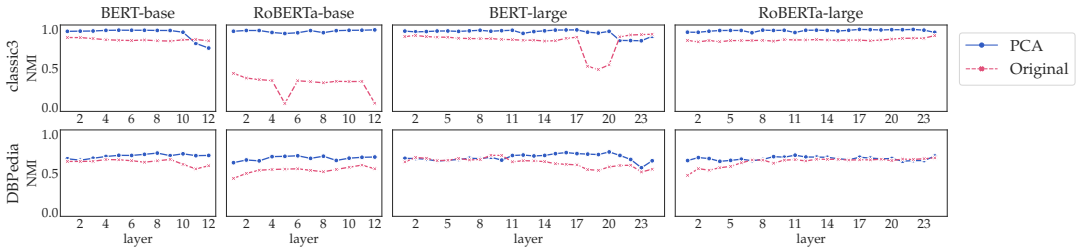

Figure 2: Clustering performance (NMI) across layers of the original representations of pre-trained models (using all of the d dimensions of the $\mathbf{X}^{(\ell)}$ matrices) compared to reduced representations ($d' = 100$), with $\ell = 1, \ldots, b$.

Table 2: NMI scores of multi-layer clustering techniques over the four Transformer models on document clustering. `LW` (layer-wise) corresponds to the mean of the NMI scores obtained by each layer ℓ (using $\mathbf{X}^{(\ell)}$) and the value between brackets to the score obtained by the best layer, a layer that unfortunately can't be identified in the absence of labels.

Dataset	Model	BOW	LW mean(best)		2nd-to-last		4 last layers (raw)			All layers (raw)			4 last layers (PCA)			All layers (PCA)		
			Raw	PCA	Raw	PCA	AVG	Concat	ENS	AVG	Concat	ENS	AVG	Concat	ENS	AVG	Concat	ENS
classic3	BERT-base	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-large		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-base		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-large		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-base	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-large		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-base		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-large		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-base	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-large		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-base		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-large		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-base	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-large		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-base		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-large		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-base	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-large		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-base		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-large		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
Mean rank (lower → better)		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**

3 CLUSTERING WITH AN ESTIMATED NUMBER OF CLUSTERS

All the results presented previously are based on the assumption that the exact number of clusters is known in advance, which is not realistic in real life. In contrast, another significant advantage offered by the proposed approach is the use of an algorithm for which the number of clusters is not known *a priori* (Louvain in our experiments). This means that the consensus returns a partition with clusters that respect the original cluster assignments as much as possible, without necessarily providing the same number of clusters as the input partitions. It is therefore a good alternative when the exact number of clusters is unknown. Also, the number of clusters for each partition is not necessarily the same, which is an interesting feature of the ensemble clustering. In order to benefit from this property, given a set $\mathcal{K}$ of some selected values of k, we ensure that each K-means run takes as input a value $k \in \mathcal{K}$ while covering as much as possible the whole set of values. We use in our experiments $\mathcal{K} = [k_r - 5, k_r + 5]$ where k_r is the real number of clusters, e.g. for the DBPedia dataset where $k_r = 14$ using a "base" model with 12 layers, the list of the 12 values of k could possibly be $\{9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 9\}$, obtaining 12 partitions $\mathbf{p}^{(1)}, \ldots \mathbf{p}^{(\ell)}, \ldots \mathbf{p}^{(12)}$ from which we compute the consensual partition $\mathbf{p}^*$ as described in Algorithm 1. The $\mathbf{p}^*$ partition then groups in the same cluster the individuals that are usually grouped together in the input partitions without having the constraint of a fixed k. This guarantees a robust clustering performance while automatically estimating the number of clusters. Table 3 shows the results of the ensemble algorithm (using the compressed representations) obtained with varying $k \in [k_r - 5, k_r + 5]$ (k_r is given in Table 1).

Table 3: Performance obtained by clustering ensemble using all the layers with an estimated number of clusters.

Model	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_b	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_ℓ	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_b	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_ℓ	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

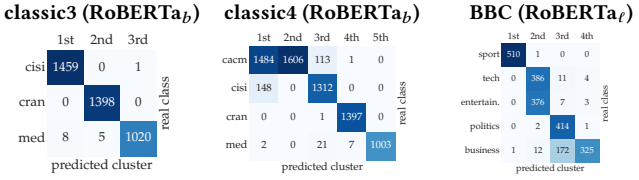

Figure 3: Confusion matrices obtained by clustering with an estimated number of clusters (as described in Section 3).

We observe from Table 3 that the performance is not significantly altered, even when the estimated number of clusters is not equal to k_r. For classic3, classic4 and AG-news, the clustering ensemble with varying k always finds k_r clusters, except for classic4 using

RoBERTa-base, where a partition of 5 clusters is found but with a high NMI, which is explained by the fact that the extra cluster corresponds to the split of the "cacm" class (cf. Figure 3). On the contrary, the number of clusters is underestimated for the BBC dataset (4 clusters instead of 5), for which the classes "tech" and "entertainement" seem to be merged, as well as the "politics" class and some "business" examples. For DBPedia, the representations provided by RoBERTa-base are the ones presenting an estimation of k that is the closest to k_r, along with a high NMI score. In that case, as for BBC, some classes are merged together such as "animal" with "plant" and "film" with "written work".

4 CONCLUSION

We have studied the performance of embeddings obtained from four pre-trained Transformer models when used as input to a document clustering algorithm. This is a contribution to the use of Transformer representations in the unsupervised learning setting.

Our experiments show that the proposed clustering ensemble method combined with PCA-based reduction allows to make the most of Transformer-based models, achieving even better performance than that provided by the best layer (which is very difficult to identify in an unsupervised context). Paths for future research include continuing to improve the ensemble procedure, especially the estimation of the number of clusters and experimenting with other dimension reduction techniques. Another perspective is evaluating the impact of fine-tuning Transformer models on text clustering.

REFERENCES

[1] Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. 2020. Ensemble block co-clustering: a unified framework for text data. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 5–14.

[2] Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. 2020. Spectral clustering via ensemble deep autoencoder learning (SC-EDAE). *Pattern Recognition* 108 (2020), 107522.

[3] Danielle S Bassett, Mason A Porter, Nicholas F Wymbs, Scott T Grafton, Jean M Carlson, and Peter J Mucha. 2013. Robust detection of dynamic community structure in networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 23, 1 (2013), 013142.

[4] Vladimir Berikov and Igor Pestunov. 2017. Ensemble clustering based on weighted co-association matrices: Error bound and convergence properties. *Pattern Recognition* 63 (2017), 427–436.

[5] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[6] Christian Buchta, Martin Kober, Ingo Feinerer, and Kurt Hornik. 2012. Spherical k-means clustering. *Journal of statistical software* 50, 10 (2012), 1–22.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[8] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*. Springer, 1–15.

[9] Derek Greene and Pádraig Cunningham. 2006. Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*. ACM Press, 377–384.

[10] Yaru Hao, Li Dong, Furu Wei, and Ke 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)*. Association for Computational Linguistics, Hong Kong, China, 4141–4150. https://doi.org/10.18653/v1/D19-1424

[11] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2, 1 (1985), 193–218.

[12] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna 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)*. Association for Computational Linguistics, Hong Kong, China, 4364–4373. https://doi.org/10.18653/v1/D19-1445

[13] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web* 6, 2 (2015), 167–195.

[14] Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic Knowledge and Transferability of Contextual Representations. In *Proceedings of the 2019 Conference of the North*. Association for Computational Linguistics, Minneapolis, Minnesota, 1073–1094. https://doi.org/10.18653/v1/N19-1112

[15] 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 preprint arXiv:1907.11692* (2019).

[16] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Oakland, CA, USA, 281–297.

[17] Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-Top: Simple and Effective Postprocessing for Word Representations. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HkuGJ3kCb

[18] Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting Contextual Word Embeddings: Architecture and Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 1499–1509. https://doi.org/10.18653/v1/D18-1179

[19] Vikas Raunak, Vivek Gupta, and Florian Metze. 2019. Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. 235–243.

[20] Nils Reimers and Iryna 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)*. Association for Computational Linguistics, Hong Kong, China, 3980–3990. https://doi.org/10.18653/v1/D19-1410

[21] Douglas Steinley. 2004. Properties of the Hubert-Arable Adjusted Rand Index. *Psychological methods* 9, 3 (2004), 386.

[22] Alexander Strehl and Joydeep Ghosh. 2002. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research* 3, Dec (2002), 583–617.

[23] Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 4593–4601. https://doi.org/10.18653/v1/P19-1452

[24] Betty van Aken, Benjamin Winter, Alexander Löser, and Felix 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*. ACM, Beijing China, 1823–1832. https://doi.org/10.1145/3357384.3358028

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

[26] Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing Pretrained Language Models for Lexical Semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 7222–7240. https://doi.org/10.18653/v1/2020.emnlp-main.586

[27] Han Xiao. 2018. bert-as-service. https://github.com/hanxiao/bert-as-service.

[28] Wei Xu, Xin Liu, and Yihong 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*. 267–273.

[29] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SkeHuCVFDr