Contextual Word Embeddings Clustering Through Multiway Analysis: A Comparative Study

Mira Ait-Saada, Mohamed Nadif

Contextual Word Embeddings Clustering through Multiway Analysis: A Comparative Study

Mira Ait-Saada[1,2] and Mohamed Nadif[1]

[1] Centre Borelli UMR 9010, Université Paris Cité, 75006 Paris
[2] Groupe Caisse des Dépôts, 75013 Paris
firstname.lastname@u-paris.fr

Abstract. Transformer-based contextual word embedding models are widely used to improve several NLP tasks such as text classification and question answering. Knowledge about these multi-layered models is growing in the literature, with several studies trying to understand what is learned by each of the layers. However, little is known about how to combine the information provided by these different layers in order to make the most of the deep Transformer models. On the other hand, even less is known about how to best use these modes for unsupervised text mining tasks such as clustering. We address both questions in this paper, and propose to study several multiway-based methods for simultaneously leveraging the word representations provided by all the layers. We show that some of them are capable to perform word clustering in an effective and interpretable way. We evaluate their performances across a wide variety of Transformer models, datasets, multiblock techniques and tensor-decomposition methods commonly used to tackle three-way data.

Keywords: Word Embeddings, Word Clustering, Multiway Analysis.

1 Introduction

Transformer-based contextual word embedding models have revolutionized the NLP state of the art. When fed with a word sequence, a Transformer model produces several different embeddings (one at each layer of the network) for each token in the sequence. Thus, in a word clustering context, for a dataset with n words and a model with b layers and latent dimension d, we can form a 3-way array of shape $n \times b \times d$ (Figure 1) where each word is represented by b vectors.

The information captured at the different layers greatly varies: typically, the early layers may encode local syntactic phenomena while more complex semantic aspects are captured at the higher layers [1]. As a consequence it is no surprise that using as input the embeddings produced at different layers may result in very different performance levels on a given downstream task.

This problem can be overcome in a supervised context, where it is always possible to determine the layer that maximises a certain metric. In contrast, in the unsupervised setting it is difficult to determine the best layer in advance with no a *priori* information about the data. In the absence of ground truth, since there

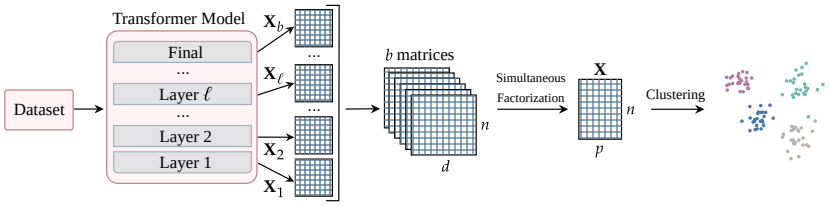

Fig. 1: Description of the proposed approach that leverages the whole Transformer model by performing multiway clustering using all of the layers' representations.

is no easy way of knowing which layer has given the best result, the solution is to design a technique guaranteeing a performance level better than that provided by the best layer, or at worst better than the mean of all layers's scores. More specifically, as a word described by several embedding vectors can be regarded as an input data observation described by several sets (blocks) of variables (dimensions), we propose to evaluate multiblock techniques [2], commonly used in chemiometrics and bioinformatics, in order to make the best use of information provided by the different layers in a word clustering context. Also, since the different layers can be seen as the slices of a 3D-tensor, we can see how do the tensor-decomposition (`TD`) techniques [3] behave.

We measure the results of these mutiblock- and tensor-based approaches on a word clustering task, and compare them with the performance obtained either with other ways of aggregating the information contained in all the layers such as *unfolding* (or *matricization*) or with layer-wise, non-multiway clustering methods where each layer is handled separately. To the best of our knowledge, this is the first study that focuses on multiway analyses to make the best use of Transformer-based word representations. The main contributions of the paper are as follows:

- While the performance delivered by Transformer embeddings is mostly assessed in the context of supervised tasks, we study the behavior of these embeddings in the unsupervised setting, and provide a thorough investigation into how to make the best use of them for word clustering purposes, across a variety of Transformer models.
- We show that certain multiway methods, simultaneously considering the features provided by all the embedding layers, are capable of delivering a good performance.

Difference with previous contributions. In previous studies, we combined clustering and Transformer representations to find meaningful groups in text datasets. In [4], we first propose a study in order to gain insight about the black-box Transformer models using unsupervised techniques including clustering. To this end, we compare layers with each other and show several differences and similarities between layers of the same model. In the present study, we go further and show the complementarity of the different layers by combining them to perform the clustering task. In [5], we propose a framework to harness the whole set of b representations provided by a Transformer model using a clustering ensemble

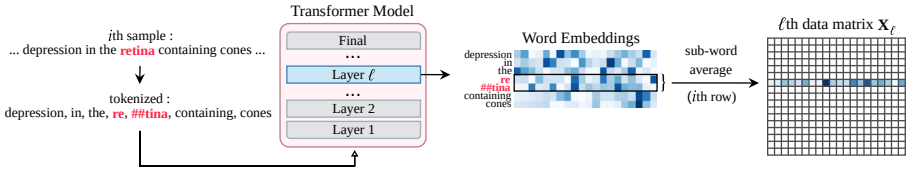

Fig. 2: Construction of the data matrix $\mathbf{X}_\ell$ from the contextual word embeddings provided by the ℓth layer. The context of the word "retina" is used to compute its embedding, which constitutes the ith sample vector representation in the $\mathbf{X}_\ell$ data matrix w.r.t. to the layer ℓ.

approach combined to a linear dimension reduction based on PCA and a whitening step. The ensemble approach consists in combining the b partitions obtained using each of the *document-level* representations. In contrast, in the present study, we carry out *word-level* clustering and more importantly, we combine the representations *before* computing the partitions, via multi-way techniques. Finally, in [6], we also perform *document* clustering and propose a graph-based three-way approach that simultaneously leverages several document representations such as document-term, static word representations and Transformer representations. In the current study, we rather leverage the representations provided by one given model and we use the representations in their raw form instead of representing then using an adjacency matrix [6].

2 Word Clustering Using Transformer Embeddings

For a given dataset of n words, and a model with b embedding layers, we obtain b different raw representations of size d, one from each layer. 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 2.

One can then choose to use each of these matrices individually, the problem being that in an unsupervised context it is not possible to determine in advance which matrix is likely to help achieve the best performance. One alternative is to try to benefit from the information provided by all the layers, simultaneously.

Using each layer's output separately result in b different clustering partitions. As already said, considering the unsupervised setting of our task and the absence of true labels, it is impossible to determine in advance the best performing layer. In addition, it is worth noting, that it is impossible to determine a unique layer for all datasets, since the best layer highly depends on the dataset used, and there is no satisfying universal choice. Therefore, we propose an alternative that benefits from all the network's layers, which we show to be very efficient while freeing us from the impossible task of choosing a unique layer to perform clustering.

Layer-wise Clustering. In layer-wise clustering, we use the K-means algorithm with as input a matrix $\mathbf{X}_\ell$ of n rows provided by the ℓth layer of the model. We also perform clustering on the `PCA`-reduced representations, that are formed by the p first principal components of $\mathbf{X}_\ell$. Layer-wise clustering is referred to as `LW`.

Unfolding Layers' Representations. To try to unleash the potential of all the layers, a simple approach consists in concatenating the matrices $\mathbf{X}_\ell, \ell = 1, \ldots, b$ into a unique data matrix $\mathbf{X}$ of size $n \times (b \times d)$. For example, given a base model with 12 layers and 768 dimensions which provides 12 matrices $\mathbf{X}_b$ of size $n \times 768$, we obtain after the unfolding a matrix of size $n \times 9\,216$. We call `UN` the use of $\mathbf{X}$ directly by a clustering algorithm.

Multiblock Analysis. With a view to taking advantage of each layer of a given Transformer language model, we propose to harness multiblock methods: Consensus PCA (`CPCA`) [7], Generalized Canonical Correlation Analysis (`GCCA`) [8], Multiple Co-Inertia Analysis (`MCIA`) [9], Multiple Factor Analysis (`MFA`) [2, 10], `STATIS` [11], Common Components and Specific Weights Analysis (`CCSWA`) [12]. These methods are designed to deal with simultaneous dimensionality reduction in b blocks (with different features describing the same observations) and are particularly popular in biological multiple omics data analysis. Thereby we argue that they can be very useful to tackle word clustering from three-way data.

In our case, we consider each layer ℓ as a block and each corresponding data matrix is represented by $\mathbf{X}_\ell$ of size $n \times d$ and $\mathbf{S}_\ell = \mathbf{X}_\ell \mathbf{X}_\ell^T$, where n is the number of samples in the dataset and d the number of features of the word representations (d is the same for all of the layers). The objective is to represent the b blocks by a unique matrix $\mathbf{W}$ of size $n \times p$ formed by p component vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_p$ each one of size $n \times 1$ that optimally resumes the overall information present in the blocks. The `MFA` method can be seen as an extension of `PCA` adapted to multiblock data. It consists in applying a standard `PCA` on a data matrix $\mathbf{X}$ whose features are composed of all the variables (dimensions) weighted according to the block (layer) they belong to in order to balance the influence of each block of variables. The balance is achieved by normalizing each data block $\mathbf{X}_\ell$ using the first eigenvalue λ_1^ℓ. $\mathbf{X}$ is then obtained by concatenating the b resulting matrices. Another formulation of the problem is finding the vector $\mathbf{q}_1$ that is the most linked to all the weighted variables. More formally, `MFA` maximizes:

$$\sum_{\ell=1}^{b} \mathcal{L}_\ell(\mathbf{X}_\ell, \mathbf{q}_1) = \sum_{\ell=1}^{b} \frac{1}{\lambda_1^\ell} \sum_{j=1}^{d} cov^2(\mathbf{x}_{\ell_{.j}}, \mathbf{q}_1) \tag{1}$$

where $\mathbf{x}_{\ell_{.j}}$ corresponds to the jth variable of the data block $\mathbf{X}_\ell$ and $\mathbf{q}_1$ to the first component of `MFA`. The next component maximizes the same criterion while being orthogonal to $\mathbf{q}_1$.

The `STATIS` method is similar to `MFA`, and differs in the weighting step. With `STATIS`, instead of weighting the data blocks according to their corresponding eigenvalues, each data block is weighted using the first eigenvector $\mathbf{v}_1$ of the inner product matrix C of size $b \times b$ where each element $c_{\ell,\ell'}$ is computed as follows:

$$c_{\ell,\ell'} = trace(\mathbf{S}_\ell \times \mathbf{S}_{\ell'}) = \sum_{i=1}^{n} \sum_{k=1}^{n} s_{i,k,\ell} s_{i,k,\ell'}. \tag{2}$$

The weighting vector $\boldsymbol{\alpha}$ is computed by $\boldsymbol{\alpha} = \mathbf{v}_1 \times (\mathbf{v}_1^T \mathbb{1})^{-1}$ so that the elements of $\boldsymbol{\alpha}$ sum to one ($\mathbb{1}$ being the unit vector). In [13], the authors discuss the

Algorithm 1: Multiway Clustering Procedure

Input: a dataset $\mathcal{D}$ of n words; a Transformer model $\mathcal{M}$ of b layers; a clustering algorithm $\mathcal{C}$; a multiway decomposition function $\mathcal{F}$, number of components p
Output: a clustering partition $\mathbf{p}$
Step 1. Build data matrices, for each $\ell = 1 \dots b$:
$\mathbf{X}_\ell \leftarrow \mathcal{M}(\mathcal{D}, \ell)$ as in Figure 2;
Step 2. Perform multiblock factorial decomposition:
$\mathbf{q}_1, \dots, \mathbf{q}_p \leftarrow \mathcal{F}([\mathbf{X}_1, \dots, \mathbf{X}_b], p)$;
$\mathbf{W} \leftarrow$ horizontal stacking of the p components $\mathbf{q}_1, \dots, \mathbf{q}_p$;
Step 3. Perform clustering:
$\mathbf{p} \leftarrow \mathcal{C}(\mathbf{W})$;
return p

difference between `CPCA`, `MCIA` and `GCCA` and show their similarity, which lies in the optimized criterion that aims to reveal covariant patterns in and between the different blocks by finding scores vectors $\mathbf{q}_1^\ell$ for each block ℓ which are as much linked as possible to a global score vector $\mathbf{q}_1$: $\sum_{\ell=1}^{b} cov^2(\mathbf{q}_1^\ell, \mathbf{q}_1)$ where $\mathbf{q}_1^\ell$ and $\mathbf{q}_1$ are $n \times 1$ vectors. This criterion is maximized by the three multiblock methods, with different constraints. The same function is maximized to find the higher order components, using a deflated version of the current data blocks at each iteration. The deflating function used with `CPCA` is as follows:

$$\mathbf{X}_\ell^{(t+1)} = \mathbf{X}_\ell^{(t)} - \frac{\mathbf{q}_t \mathbf{q}_t'}{\mathbf{q}_t' \mathbf{q}_t} \mathbf{X}_\ell. \tag{3}$$

The `CCSWA` [12] method also computes the components iteratively. The first component $\mathbf{q}_1$ and their weights $\gamma_1^1, ..., \gamma_1^b$ are computed as to minimize the expression: $\sum_{\ell=1}^{b} \left\| \mathbf{S}_\ell - \gamma_1^\ell \mathbf{q}_1 \mathbf{q}_1^T \right\|^2$ where γ_1^ℓ represents the salience value of the ℓth block for the first component (common axis) represented by the vector $\mathbf{q}_1$. The `CCSWA` algorithm aims to find the parameters $\gamma_t^1, ..., \gamma_t^b$ and $\mathbf{q}_t$ for $t = 1, 2, \dots, p$ that maximizes an overall loss function at each iteration t:

$$\sum_{\ell=1}^{b} ||\mathbf{S}_\ell - \sum_{k=1}^{t} \gamma_1^\ell \mathbf{q}_k \mathbf{q}_k^T||^2. \tag{4}$$

This proportion belongs to $[0, 1]$ and the closer to 1 it is, the better is $\mathbf{X}_{\ell'}$ as a substitute for $\mathbf{X}_\ell$ (and *vice-versa*) to characterize the n samples of the dataset.

Whatever the method used, the p first common components $\mathbf{q}_1, \dots, \mathbf{q}_p$ are used as input to a clustering algorithm as shown in Algorithm 1, thus leveraging all of the word representations provided by the multi-layered Transformer models. The common components can be obtained using the R package *FactoMineR* [14] for `MFA` and *mogsa* [15] for `CPCA`, `GCCA`, `MCIA`, and `STATIS`. For `CCSWA`, we used our Python implementation.

Tensor Decomposition. In this family of methods, data matrices $\mathbf{X}_\ell, \ell = 1, \dots, b$ can be viewed as a three-way or three-modal tensor $\mathbf{X}$ of size $n \times d \times b$ where

each matrix $\mathbf{X}_\ell$ (obtained with the ℓth layer of the model) is considered as a slice of the tensor. Two of the most popular TD techniques are CANDECAMP/PARAFAC (CP) [16] and Tucker decomposition [17]. For the sake of simplicity, we describe these two techniques for three-mode tensors, but they can be generalized for m-mode tensors ($m > 3$). A CP decomposition of rank p aims to find three factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ of size $n \times p$, $d \times p$ and $b \times p$ respectively, and $\boldsymbol{\lambda}$ that minimizes the approximation error of $\mathbf{X}$ by finding:

$$\min_{\mathbf{X}^*} \|\mathbf{X} - \mathbf{X}^*\| \text{ where } \mathbf{X}^* = \sum_{j=1}^{p} \boldsymbol{\lambda}_j \mathbf{a}_j \circ \mathbf{b}_j \circ \mathbf{c}_j \tag{5}$$

where $\mathbf{a}_j$, $\mathbf{b}_j$ and $\mathbf{c}_j$ being the jth column of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ respectively. The "$\circ$" stands for the outer product between two vectors. The outer product of the three vectors $\mathbf{a}_j$, $\mathbf{b}_j$ and $\mathbf{c}_j$ results in a three-way tensor with the sames dimensions as $\mathbf{X}$. One of the most popular procedures to optimize this criterion is the Alternating Least Squares (ALS) [16,18]. The Tucker decomposition also computes three factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, this time of size $n \times p$, $d \times q$ and $b \times t$ respectively. Unlike in CP, the ranks of the three modes (p, q and t) are not necessarily equal. In addition to the factor matrices, a core tensor $\mathbf{G}$ of size $p \times q \times t$ is computed so as to optimize:

$$\min_{\mathbf{X}^*} \|\mathbf{X} - \mathbf{X}^*\| \text{ where } \mathbf{X}^* = \sum_{i=1}^{p} \sum_{j=1}^{q} \sum_{k=1}^{t} g_{ijk}\, \mathbf{a}_i \circ \mathbf{b}_j \circ \mathbf{c}_k \tag{6}$$

g_{ijk} being the intersection if the ith row (mode 1), the jth column (mode 2) and the kth tube (mode 3) of the tensor $\mathbf{G}$. To solve this optimization problem, a popular approach named Higher-Order Orthogonal Iteration (HOOI) [19] and similar to ALS consists in fixing two factor matrices to compute the third one by using two-way singular vector decomposition (cf. [19] for further details).

In our experiments, we use for both CP and Tucker the matrix $\mathbf{A}$ as input to the clustering algorithm. The p columns of $\mathbf{A}$ can be seen as the principal components in the first mode of $\mathbf{X}$. We use two Python implementations of ALS and HOOI to perform CP and Tucker decomposition respectively. The first implementation is provided by the *TensorD* [20] package, based on the *TensorFlow* framework, allowing GPU acceleration. The second is available in the *TensorLy* [21] package which flexibly leverages several CPU and GPU backends. The suffixes TensD and TensL in Table 2 stand for the packages *TensorLy* and *TensorD* respectively.

3 Experimental Study

Datasets and Models Used. The datasets used are described in Table 1 and contain words, their contexts and the corresponding topic. The first dataset, denoted as UFSAC3 is extracted from the UFSAC dataset [22] which consolidates multiple popular WSD datasets (such as SemEval and SensEval) into a uniform format. The examples are manually divided into three topics: Body, Botany and

Geography. The second dataset, denoted as UFSAC4, is a slightly more difficult dataset; it includes a fourth class (words related to information technology) and is augmented with some polysemous examples (such as "lobes" which appears in Body and Botany with different contexts). The two last datasets yahoo4 and yahoo6, are extracted from the Yahoo! Answers dataset [23] by manually selecting sets of words for each category and some corresponding contexts.

Table 1: Datasets description: k denotes the number of clusters.

datasets	n	k	clusters' sizes
UFSAC3	583	3	body: 266, geo: 227, botany: 90
UFSAC4	691	4	body: 275, geo: 227, botany: 99, IT:90
yahoo4	528	4	finance: 150, music: 82, science-maths.: 152, computers-internet: 144
yahoo6	852	6	health: 201, finance: 150, computers-internet: 144, music: 82, science-maths.: 152, sports: 123

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 [24] and RoBERTa [25] with a vocabulary size of 28,996 for BERT and 50,265 for RoBERTa.

Clustering Evaluation. In all the experiments, we use K-means [26] with a known number of clusters (any other clustering algorithm can be used), which are run with 10 different initializations on different matrix representations of words occurrences. To validate the results produced by the clustering algorithm we relied on standard external measures devoted to assessing cluster quality, namely Normalized Mutual Information (NMI) [27], the Adjusted Rand Index (ARI) [28] and the clustering accuracy. When performing dimension reduction, a number of components p has to be fixed. Since we cannot tune the p parameter in our unsupervised setting, we set it to the same value ($p = 15$) for all experiments. This corresponds either to the number of principal components of `PCA`, the number of common components present in the $\mathbf{W}$ matrix computed by the multiblock techniques or the rank of the matrix $\mathbf{A}$ computed by the `TD` methods (cf. Section 2). Table 2 contains the NMI scores obtained by each method over the four Transformer models and the four datasets. For comparison purposes, we also include the results obtained when using fastText word embeddings or the representations obtained by the Transformers' layers individually (as described in the next section). FastText [29] was used as a representative of the static word embeddings family since it provides for the Out-of-Vocabulary issue which otherwise would distort comparisons.

Layer-wise Clustering Experiments. We perform clustering on the `PCA`-reduced representations and present the results (NMI) in Figure 3 only for two datasets, due to the lack of space. In Table 2, we call `LW` and `LW-PCA` the average of the scores obtained by all of the b clustering experiments run on each matrix $\mathbf{X}_\ell$ separately before and after applying `PCA` respectively.

Table 2: Compared performance (NMI scores) of multiway clustering on Transformers' over the four models and fastText. Values between parentheses correspond to the standard deviation.

Datasets	Method	BERT-base	BERT-large	RoBERTa-base	RoBERTa-large	fastText
UFSAC3	LW	0.89 (0.07)	0.81 (0.3)	0.71 (0.14)	0.64 (0.16)	0.92
	LW-PCA	0.91 (0.07)	0.8 (0.31)	0.78 (0.16)	0.66 (0.16)	0.92
	UN	0.96	**1.0**	0.94	0.66	-
	UN-PCA	**0.98**	**1.0**	**0.97**	0.65	-
	MFA	**0.98**	**1.0**	**0.97**	0.77	-
	STATIS	0.96	0.98	**0.97**	0.77	-
	CPCA	0.76	0.71	0.68	0.53	-
	GCCA	0.8	0.76	0.74	0.65	-
	MCIA	0.7	0.76	0.72	0.69	-
	CCSWA	0.58	0.58	0.77	0.68	-
	CP-TensD	0.66	0.28	0.01	0.64	-
	CP-TensL	0.92	0.66	0.57	0.56	-
	Tuck-TensD	0.68	0.75	0.78	0.83	-
	Tuck-TensL	0.73	0.81	0.89	0.69	-
UFSAC4	LW	0.86 (0.08)	0.72 (0.28)	0.67 (0.11)	0.6 (0.13)	0.77
	LW-PCA	0.88 (0.06)	0.75 (0.25)	0.74 (0.1)	0.63 (0.11)	0.84
	UN	0.96	0.97	0.67	0.65	-
	UN-PCA	**0.97**	**0.99**	0.76	0.65	-
	MFA	**0.97**	**0.99**	0.73	0.67	-
	STATIS	0.96	0.95	**0.90**	0.68	-
	CPCA	0.75	0.84	0.75	0.59	-
	GCCA	0.76	0.75	0.82	0.7	-
	MCIA	0.81	0.76	0.79	0.67	-
	CCSWA	0.73	0.82	0.76	**0.76**	-
	CP-TensD	0.43	0.59	0.06	0.46	-
	CP-TensL	0.61	0.55	0.48	0.61	-
	Tuck-TensD	0.75	0.85	0.84	0.62	-
	Tuck-TensL	0.81	0.86	0.79	0.65	-
yahoo4	LW	0.81 (0.07)	0.59 (0.24)	0.54 (0.17)	0.59 (0.14)	0.42
	LW-PCA	0.83 (0.07)	0.58 (0.25)	0.84 (0.04)	0.66 (0.15)	0.51
	UN	0.9	0.82	0.63	0.71	-
	UN-PCA	**0.93**	**0.91**	**0.91**	0.82	-
	MFA	0.92	**0.91**	**0.91**	**0.83**	-
	STATIS	0.91	0.89	0.9	0.82	-
	CPCA	**0.93**	0.74	0.87	0.82	-
	GCCA	0.88	0.71	0.85	0.79	-
	MCIA	0.85	0.76	0.86	0.8	-
	CCSWA	0.7	0.73	0.81	0.8	-
	CP-TensD	0.04	0.08	0.17	0.07	-
	CP-TensL	0.44	0.55	0.41	0.63	-
	Tuck-TensD	0.81	0.66	0.81	0.79	-
	Tuck-TensL	0.83	0.73	0.85	0.8	-
yahoo6	LW	0.73 (0.08)	0.58 (0.21)	0.61 (0.15)	0.62 (0.16)	0.42
	LW-PCA	0.73 (0.06)	0.57 (0.22)	0.79 (0.08)	0.69 (0.14)	0.42
	UN	0.92	0.8	0.72	0.79	-
	UN-PCA	0.94	0.81	0.81	0.81	-
	MFA	0.93	0.83	0.81	0.81	-
	STATIS	0.74	0.75	0.8	0.81	-
	CPCA	0.94	**0.9**	0.88	**0.84**	-
	GCCA	**0.96**	0.85	0.88	0.83	-
	MCIA	0.93	0.86	**0.9**	0.83	-
	CCSWA	0.82	0.7	0.65	0.76	-
	CP-TensD	0.53	0.18	0.09	0.11	-
	CP-TensL	0.6	0.74	0.57	0.46	-
	Tuck-TensD	0.87	0.71	0.78	0.81	-
	Tuck-TensL	0.85	0.66	0.79	0.81	-

Multiway Clustering Experiments. Since the obtained results may greatly vary with each layer, as clearly visible in Figure 3, and since determining which one is the best is impossible in the absence of labels when dealing with real datasets, a sensible solution to overcome this incertitude is to use multiway techniques as described in Section 2 using the $\mathbf{X}_\ell$ data matrices $\ell = 1, \ldots, b$ of each model (results provided in Table 2).

4 Results and Discussion

In this section, we discuss the benefit of applying multiway methods in the unsupervised setting due to a diversity of the information provided by each layer. Thereby, from the representations of each layer used separately we first evaluate the performance in terms of word clustering. We aim at showing that the results might greatly vary according to the dataset.

To tackle this issue, for each dataset we propose to use all layers. We perform a dimensionality reduction in different ways and from a simplified representation, a clustering algorithm is applied and quality of word clustering is measured. Further, using user-friendly visualization tools, we show how the classes are arranged in a low-dimensional space, how to detect the most influential words and to measure the role of each layer.

Layer-wise Clustering Results. Figure 3 shows that BERT presents better performance than RoBERTa, even if the latter is more recent and outperforms BERT in the supervised tasks. We can see in the same figures that BERT models achieve their best performance on the intermediate layers (which is in line with the findings issued in [30]) especially BERT-large which reaches a perfect NMI score on UFSAC3 from the 9th to the 14th layer, with a sudden drop of performance on the 5 last layers. Overall, we can see this decrease for all the models. Figure 3 also shows that reducing the number of dimensions to only 15 (which constitutes less than 2% of features for the base models and 1.5% for the large ones) leads to very competitive performance scores, achieving even better results than raw representations, which indicates a high redundancy in the pre-trained embeddings. Interestingly, in the case of RoBERTa the `PCA` reduction generally improves the performance of the last layers, which as said previously perform very poorly in their raw form.

Multiway Clustering Results. One first observation from Table 2 concerning the multiway clustering results is that multiblock methods seem more effective compared to `TD` techniques. Tucker seems more suitable for our data than `CP`. Besides, Table 2 shows that in the case of BERT (base and large), the multiblock techniques are very competitive. Moreover, `MFA` and `UN-PCA` on BERT-large have no difficulty in achieving the perfect score on UFSAC3. For RoBERTa-large, multiblock techniques are still better than the mean of all layers's scores. In general, the performance of `MFA` is much higher than the average performance of the layers used separately. This indicates that it is very effective for capturing

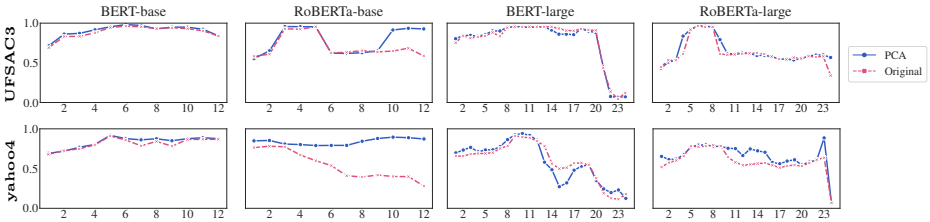

Fig. 3: Compared clustering performance (NMI) across layers of the original representations of base models (using all of the 768 dimensions of the $\mathbf{X}_\ell$ matrices) vs. reduced representations (using 15 principal components of $\mathbf{X}_\ell$), with $\ell = 1, \ldots, b$ ($b \in \{12, 24\}$).

the valuable information present in the $d \times b$ features of each model (9 216 for the base models and 24 576 for the large ones) in only 15 dimensions.

In Table 2 we observe that the raw unfolding (`UN`) is less powerful than when reduced with `PCA` (`UN-PCA`). This brings out the interest of factor decomposition and dimension reduction in improving performance in the clustering task while combining all layers. This difference in performance is in line with the observations made on Figure 3 where the use of `PCA` noticeably improved the layer-wise clustering, which explains the enhancement observed with `UN-PCA`.

Visual Interpretation of Factorial Analysis Results. One additional advantage of Multiblock methods and specifically `MFA` is that they offer several visual interpretation possibilities that are presented in Figures 4, 5 and 6.

Figure 4 illustrates the words' projections on the two first common factors of `MFA` from two datasets. We observe a good separation of the three clusters of UFSAC3, especially with BERT-large, which is in line with the scores provided in Table 2.

Figure 5 shows the contribution of the first 20 words to the first three components of `MFA` for BERT-base and RoBERTa-base. We see that for BERT-base, among the 20 most contributory words, the classes are perfectly distributed over the 3 first dimensions (Geography, Botany and Body contribute the most to dimensions 1, 2 and 3 respectively). For RoBERTa-base, the distribution is less homogeneous, the three classes contributing equally to the last dimension.

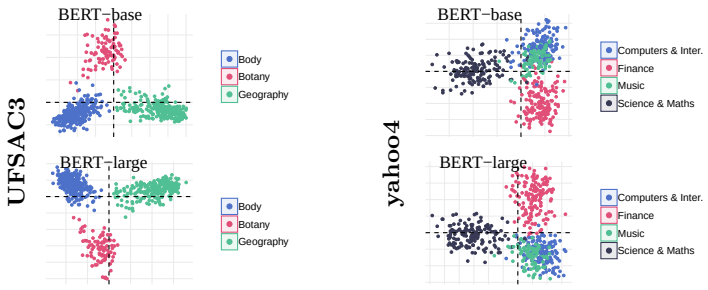

Fig. 4: 2D projections of UFSAC3 and yahoo4 words on the two first common components of `MFA`, colored according to the known topics.

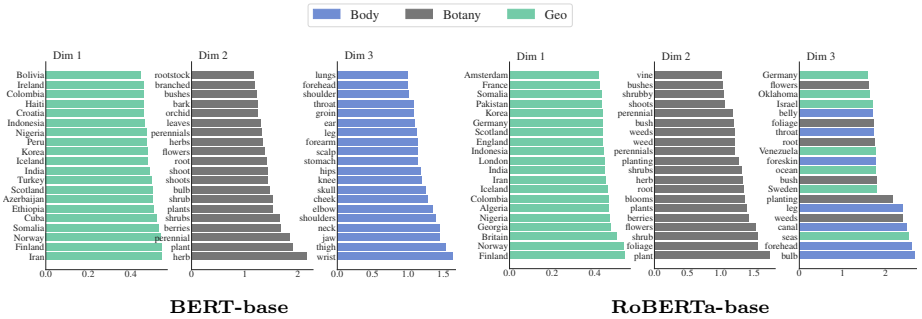

Fig. 5: The 20 most contributory words (of UFSAC3) to the first three dimensions of MFA applied to all the layers of BERT-base and RoBERTa-base.

Indices to help with interpretation. Figure 6 shows the contribution and NMI score of each layer with the 4 models. The contribution of a layer to a component is the sum of the contributions of its variables. More formally, the contribution of the ℓth layer to the jth component is computed as $\sum_{k \in \mathcal{G}_\ell} \boldsymbol{\alpha}_\ell \times \mathbf{u}_{k,j}$ where $\mathbf{u}_{k,j}$ is the loading of the kth variable for the jth component, $\boldsymbol{\alpha}_\ell = \frac{1}{\lambda_1^\ell}$ is the weight of the ℓth layer and $\mathcal{G}_\ell$ is the set of variables of the ℓth layer. The contribution of each variable takes values between 0 and 1 and sums to 1 for a given component. The values of contribution displayed in Figure 6 are percentages. We observe from Figure 6 that `MFA` seems to sum up the relevant information present in the layers of the models, without taking into account the most outlying (and less performing) layers, especially for BERT-base and BERT-large. For the other models, the drop of performance occurring at the end of some models coincides with a significant decrease of contribution. We can also see that for RoBERTa-base, the best performing layer (layer 5) is also the most contributory to the first component of `MFA`.

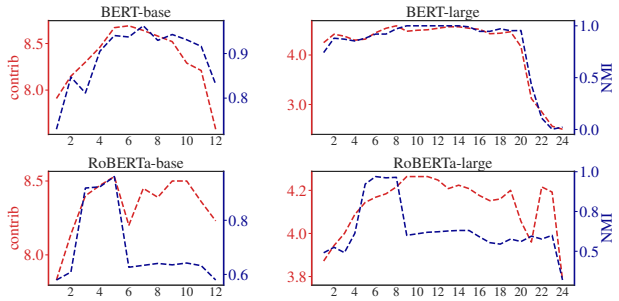

Fig. 6: The *contribution* of each block (layer) to the first common component of `MFA` (UFSAC3). The NMI (in red) of a layer ℓ is obtained with clustering separately the $\mathbf{X}_\ell$ matrix made of the original word representations.

5 Conclusion

The present study has shown that a multiway-based approach to word clustering has a twofold benefit. It first removes the need to choose, among the different layers of a Transformer, the one supposed to be the most useful input, something impossible to

determine in an unsupervised setting. Second, across a variety of Transformer models as well as datasets for which labels were available for evaluation, we have shown that multiway techniques can deliver a good performance without requiring the choice among layers, a topic widely discussed in the NLP community. Among these techniques, we retain `MFA` which turns out to be the most consensual, outscoring standard tensor-decomposition methods. This can be attributed to the fact that `MFA` aims at giving a balanced role to the layers. It is based on a factor analysis in which the variables are weighted and these weights are identical for the dimensions of the same layer (and vary from one layer to another) while offering a visualisation with many indices to help with interpretation. Thereby, we have shown for the first time the interest of using such a method in NLP.

References

1. Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy, 2019.
2. B Escofier et al. Méthode pour l'analyse de plusieurs groupes de variables. application à la caractérisation de vins rouges du Val de Loire. *Revue de statistique appliquée*, 31(2):43–59, 1983.
3. Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.
4. Mira Ait-Saada, François Role, and Mohamed Nadif. Unsupervised methods for the study of transformer embeddings. In Pedro Henriques Abreu, Pedro Pereira Rodrigues, Alberto Fernández, and João Gama, editors, *Advances in Intelligent Data Analysis XIX*, pages 287–300, Cham, 2021. Springer International Publishing.
5. Mira Ait-Saada, François Role, and Mohamed Nadif. 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 & Knowledge Management*, CIKM '21, page 2837–2841, 2021.
6. Rafika Boutalbi, Mira Ait-Saada, Anastasiia Iurshina, Steffen Staab, and Mohamed Nadif. Tensor-based graph modularity for text data clustering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2227–2231, 2022.
7. S Wold, S Hellberg, T Lundstedt, M Sjostrom, and H Wold. Proc. symp. on pls model building: theory and application. *Frankfurt am Main.*, 1987.
8. J Douglas Carroll. Generalization of canonical correlation analysis to three or more sets of variables. In *Proceedings of the 76th annual convention of the American Psychological Association*, volume 3, pages 227–228. Washington, DC, 1968.
9. Daniel Chessel and M Hanafi. Analyses de la co-inertie de k nuages de points. *Revue de statistique appliquée*, 44(2):35–60, 1996.
10. Hervé Abdi, Lynne J Williams, and Domininique Valentin. Multiple factor analysis: principal component analysis for multitable and multiblock data sets. *Wiley Interdisciplinary reviews: computational statistics*, 5(2):149–179, 2013.
11. Yves Escoufier. L'analyse conjointe de plusieurs matrices de données. *Biométrie et temps*, 58:59–76, 1980.
12. El Mostafa Qannari, Ian Wakeling, Philippe Courcoux, and Halliday JH MacFie. Defining the underlying sensory dimensions. *Food Quality and Preference*, 11(1-2):151–154, 2000.

13. Mohamed Hanafi, Achim Kohler, and El-Mostafa Qannari. Connections between multiple co-inertia analysis and consensus principal component analysis. *Chemometrics and intelligent laboratory systems*, 106(1):37–40, 2011.
14. Sébastien Lê, Julie Josse, and François Husson. FactoMineR: A package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18, 2008.
15. Chen Meng. *mogsa: Multiple omics data integrative clustering and gene set analysis*, 2019. R package version 1.20.0.
16. J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, 1970.
17. Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.
18. Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. 1970.
19. Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.
20. Liyang Hao, Siqi Liang, Jinmian Ye, and Zenglin Xu. Tensord: A tensor decomposition library in tensorflow. *Neurocomputing*, 318:196–200, 2018.
21. Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in Python. *Journal of Machine Learning Research*, 20(26):1–6, 2019.
22. Loïc Vial, Benjamin Lecouteux, and Didier Schwab. UFSAC: Unification of sense annotated corpora and tools. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.
23. Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657, 2015.
24. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019.
25. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.
26. Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.
27. Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.
28. Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.
29. Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.
30. Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. 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)*, pages 1073–1094, Minneapolis, Minnesota, June 2019.