Towards Explainable Graph Spectral Clustering for BERT Embeddings

Submitted: 10th July 2025; accepted: 10th August 2025

Mieczysław A. Kłopotek, Sławomir T. Wierzchoń, Bartłomiej Starosta, Piotr Borkowski, Dariusz Czerski

DOI: 10.14313/jamris-2026-005

Abstract:
Artificial Intelligence algorithms are increasingly applied to tasks in Natural Language Processing, including document clustering. As these algorithms become increasingly complex (such as transformer-based embeddings, like BERT) and/or are of a "black-box" nature, such as Graph Spectral Clustering (GSC) algorithms, the demand for explaining the results of such algorithms is becoming increasingly urgent. In this paper, we propose a model-aware method to explain the results of GSC in the context of BERT-based embeddings. We present a novel theoretical methodology for explanation, based on the premise that document similarity in GSC is computed as cosine similarity of BERT embeddings of documents. We demonstrate the validity of this methodology by presenting strong GSC clustering results, restoring the human-made assignment of hashtags to tweets. We show that GSC based on BERT embeddings outperforms approaches using Term Vector Space and GloVe embeddings. Therefore, the resulting explanations are also expected to be of higher quality.

Keywords: *Explainable Machine Learning, Natural Language Processing, Graph Spectral Clustering, Document Embedding versus Explainability, BERT and GloVe and TVS Embedding*

1. Introduction

As noted in [1], accessing textual information has never been easy in human history. There was a time when text documents were scarce; today, we face an overwhelming stream of diverse materials. However, in each case, creating overviews of document collections is vital. For centuries, subject indexing and classification systems have served this purpose. In the past, it was possible to prepare a list of categories and then manually add documents to that list. However, today there is a growing urgency to automate both tasks.

Although there exist elaborate human-made category dictionaries for general purposes, specialized collections of documents require specialized local methods for splitting the documents into reasonable groups and labeling them appropriately. The latter task seems to require the involvement of automated methodologies.

Clustering methodologies and accompanying technologies for explaining group membership can prove to be helpful supporting tools here.

Methods of clustering text documents have already found numerous applications. They can help uncover connections between different texts, group them into "natural" categories, or pinpoint the most relevant topics in their content and express them in distinct terms. They are used for rapid information retrieval and filtering, topic extraction, and automatic document organization.

Clustering requires the development of a similarity measure between documents. The simplest way would be to embed the documents in an Euclidean space and (1) to use the distance between the embedding points and apply common algorithms, such as k-means [2] for clustering in such a space, or (2) to use, for example, the cosine similarity between embedding vectors and then use similarity-based algorithms like those of the Graph Spectral Clustering class [3]. Although a multitude of similarity measures have been proposed [4], cosine similarity seems to reflect semantic similarity quite well [5].[1] The latter is embraced in this paper.

Many types of embedding spaces have been developed since the very first proposal of term vector space (TVS) [6], [7]. Despite its straightforwardness, TVS has two notable drawbacks: (1) the document is regarded as a collection (or "bag") of words, which leads to the loss of context and the relationships among terms, and (2) the dimensionality can be as high as dozens of thousands, even for a moderate-size collection of several thousand documents. Therefore, new embedding approaches, such as Word2vec [8], Doc2Vec [9], GloVe [10], or BERT [11], and many others (see [12–14], or `https://huggingface.co/spaces/mteb/leaderboard`), were developed to accommodate relationships between terms, and to reduce dimensionality, although dimensionality can still be high, reaching hundreds of dimensions.

Fortunately, an embedding space with a small number of dimensions can be sufficient if GSC is used for clustering.

Usually, we do not need only the clusters of textual documents but also a characterization of their content in terms of keywords, for example. The usual approach to this task was to get the clustering first, and then to seek differences/similarities in the word sets related to each of the clusters. This approach can be termed the "characterization of clusters." It identifies the features of the obtained clusters but does not explain why a document belongs to one cluster rather than another. Yet often, we want to understand why a document is part of a group and why the algorithm put it in that group. We shall term this "cluster membership explanation". It seems like a simple process if we cluster the documents directly in the TVS via, e.g., k-means algorithm. This is because the center coordinates of the clusters show how important the words/terms are for the cluster membership.

However, both when clustering in the modern embedding spaces, like word2vec, doc2vec, GloVe, BERT and other transformer methods, and when using GSC based clustering, we get results that are hard to explain as the relationship between the embedding space coordinates and document words/terms gets lost which is counterproductive with respect to the librarian's goal to assign some automated subject index. For example, in GSC, the clustering process is completely detached from the words, as only similarities are used. BERT embedding, on the other hand, makes the relationship between words and document embedding completely unrecoverable.

In this paper, we make an attempt to overcome the mentioned problems with explainability. We outline a methodology for the explanation of GSC results performed in a BERT embedding, whereby cosine similarity is used as similarity method. Before doing so in section 4, we first recall some related work. In section 2 we provide a brief overview of the methodologies behind BERT embeddings of documents. In section 3, we present an introduction to GSC methodology. In Section 5 we present experiments supporting the validity of our approach. Section 6 concludes the paper.

2. BERT Embedding Extractions

BERT (Bidirectional Encoder Representations from Transformers), available since 2018, is a method of pretraining language representations of natural language text modeling [11]. One can either use these models to extract high-quality language features from one's text data, or one can fine-tune these models on a specific machine learning (ML) task, such as classification, entity recognition, or question answering, based on one's dataset. There exist multiple BERT models nowadays, starting with the so-called `bert-base-uncased`, a 110 M parameter model containing 12 layers (blocks) and working with lowercase text.[2]

To extract the contextual embedding of each word in the sentence, one needs to tokenize the document and feed the tokens, together with positional and token type information, to the pre-trained BERT, which will return the embeddings for each of the tokens. Apart from obtaining the token-level (word-level) representation, one can also obtain the document level representation. The result is obtained by passing the input through multiple (12 in `bert-base-uncased`) neural layers. Each layer (called a transformer block) consists of two sublayers. The first sub-layer implements a multi-head self-attention mechanism, and the second one is a position-wise fully connected feed-forward network. The output of each sublayer is added to the input that bypasses the sublayer through a residual connection, and the resulting signal ends in layer normalization that is passed to the next layer. At the final layer, one gets a representation of each token at each position. Note that generally, BERT models return token embeddings relative to a given sentence/document. This means that the same word can have different embeddings in different sentences, and even in the same sentence when it is used several times.

As explained by [15], there are multiple ways of extracting embeddings of words, sentences, or documents from pre-trained BERT models.

As words are concerned, there exist multiple possibilities to obtain a static word embedding. For example, the "Averaged BERT" method consists of averaging the representations of the same word in its different contexts to acquire a static embedding. The "Averaged plus Regular BERT" approach means that the representation of a word in a given context is the sum of the original BERT representation plus the mentioned averaged value. One may not restrict oneself to the last layer only, but use, e.g., last 4 layers. Other variants are also discussed in the literature.

The simplest representation of the entire text is to use the average of the embeddings of individual words. More advanced is the weighting of words with the logarithm of inverted document frequency.

The obtained vectors may be normalized to the unit length, or via dividing by their standard deviations, and in many other ways.

BERT was used in ML applications requiring explanation. For instance, [16] utilized the BERT model for the purpose of medical classification, using a two-stage model. The first stage was used for purposes of document embedding, while the second stage was trained to predict the classification. Explanations were word-based, where the word importance for classification decision was produced based on a gradient-based method called integrated gradients [17, 18].

[19] discusses the explainability of BERT model results in the task of sentiment analysis.

3. A Short Introduction to GSC

Let us now explain how to exploit the properties of the Graph Spectral Clustering methodology when clustering documents in BERT embedding. Usage of GSC for clustering instead of direct clustering in the BERT embedding has the advantage of reducing the dimensionality of the clustering problem by an order of magnitude. As we will show subsequently, the result of GSC approximates the result of direct clustering in BERT embedding. This allows clustering in a low-dimensional space. In addition, it is possible to build an explanatory bridge to the GSC/BERT combination.

Let us characterize the GSC approach to clustering briefly. A more detailed description can be found, for example, in [3].

Let S be a (symmetric) similarity matrix between pairs of items (documents, like tweets in our case).

It induces a graph whose nodes correspond to the entities (documents), hence the "Graph" part of the GSC name. Let n denote the number of items for which S has been computed. Let D be the diagonal matrix with $d_{jj} = \sum_{k=1}^{n} s_{jk}$ for each $j \in [n]$.

In the domain of text mining, the mentioned similarity matrix is usually based on either a graph representation of relationships (links, hyperlinks) between items, or such a graph is induced by (cosine) similarity measures between these items. However, mixed object representations (text and links) have also been studied, for example by [20]. In this paper, we use the cosine similarity between BERT embedding vectors. Hence, the elements of S are of the form $s_{i\ell} = g(\delta_i)^T g(\delta_\ell)$, where $g(\delta_i)$ stands for the BERT embedding for document δ_i. We assume that all these embeddings are vectors of unit length. Additionally, by GSC convention, all diagonal elements of S are zero. (Unnormalized or) combinatorial Laplacian L corresponding to this matrix is defined as

$$L = D - S\,. \tag{1}$$

A *normalized Laplacian* $\mathcal{L}$ of the graph represented by S is defined as

$$\mathcal{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} S D^{-1/2}\,. \tag{2}$$

The *rationormalized Laplacian*[3] takes the form

$$\mathcal{L}_{\mathcal{R}} = {D'}^{-1/2} L {D'}^{-1/2} = I - {D'}^{-1/2} S' {D'}^{-1/2} \tag{3}$$

where $S' = S + I$ and $D' = D + I$.

The split into k disjoint clusters is achieved as follows. One computes the eigen-decomposition of the respective Laplacian (e.g., L, $\mathcal{L}$, or $\mathcal{L}_{\mathcal{R}}$), getting n eigenvalues $\lambda_1 \le \ldots \le \lambda_n$ (always $\lambda_1 = 0$, due to mathematical properties of the mentioned Laplacian's) and corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Then one embeds the documents into the k-dimensional space spanned by the k eigenvectors corresponding to k lowest positive eigenvalues. That is, i-th document is represented by the vector $\mathfrak{x}_i = [v_{i,2}, \ldots, v_{i,k+1}]^T$. This shall be called L-embedding, resp. N-embedding, or R-embedding if the eigenvectors are determined from the combinatorial (resp. normalized or rationormalized) Laplacian L, $\mathcal{L}$, or $\mathcal{L}_{\mathcal{R}}$. Mathematical properties imply that the eigenvector $\mathbf{v}_1$ of the combinatorial Laplacian is a constant vector; hence, to get an informative embedding, we use the eigenvectors $\mathbf{v}_2, \ldots, \mathbf{v}_{k+1}$. We also apply this convention to the other embeddings, that is N and R for consistency. Once a proper embedding has been determined, it is possible to cluster the documents in a chosen embedding space using, for example, the k-means algorithm. See [3] or [21] for details. k-means clustering in the L-embedding shall be called L-based clustering. N-based clustering and R-based clustering be defined analogously. Note also, that the L-based clustering allows us to approximate the clustering optimizing so-called RCut criterion. In contrast, the N-based clustering approximates the clustering optimizing NCut criterion, while R-based clustering allows the approximation of the NRCut clustering criterion, a mixture of both.

Note, that it is possible to formalize the three clustering criteria of the form

$$Q^{[GSA\mathfrak{X}]}(\Gamma) = \sum_{j=1}^{k} \sum_{i \in C_j} ||\mathfrak{x}_i - \mu(C_j)||^2 \tag{4}$$

$$= \frac{1}{2} \sum_{j=1}^{k} \frac{1}{|C_j|} \sum_{i \in C_j} \sum_{\ell \in C_j} ||\mathfrak{x}_i - \mathfrak{x}_\ell||^2 \tag{5}$$

where $\Gamma = \{C_1, \ldots, C_k\}$ is a partition of the set of documents, and $\mathfrak{x}_i$ is a vector with components x_{ij} ($j \in 1:k$) being the i-th entries of the first k eigenvectors of the appropriate matrix L, $\mathcal{L}$, or R. That is if $\mathbf{v}_j$ is an eigenvector, then $x_{ij} = v_{j,i}$.

The j-th cluster center is defined

$$\mu(C_j) = \frac{1}{|C_j|} \sum_{i \in C_j} \mathfrak{x}_i \tag{6}$$

where $\mathfrak{x}_i$ represents embedding of i-th document as described above.

The formula (4), is precisely the loss function of the k-means algorithm in the respective (Euclidean) embedding space. So, it is quite natural that their minima are sought by applying the traditional k-means algorithm.

Graph clustering was primarily associated with the quantity

$$cut(C_j, \bar{C}_j) = \sum_{i \in C_j} \sum_{\ell \notin C_j} s_{i\ell} \tag{7}$$

which represents the aggregate similarity of nodes that are neighbors in a given graph but belong to different clusters. Properly normalizing this quantity we obtain three different criteria, i.e.,

$$RCut(\Gamma) = \sum_{j=1}^{k} \frac{cut(C_j, \bar{C}_j)}{|C_j|} = \sum_{j=1}^{k} \frac{1}{|C_j|} \sum_{i \in C_j} \sum_{\ell \notin C_j} s_{i\ell} \tag{8a}$$

$$NCut(\Gamma) = \sum_{j=1}^{k} \frac{cut(C_j, \bar{C}_j)}{\mathcal{V}_j} = \sum_{j=1}^{k} \frac{1}{\mathcal{V}_j} \sum_{i \in C_j} \sum_{\ell \notin C_j} s_{i\ell} \tag{8b}$$

$$NRCut(\Gamma) = \sum_{j=1}^{k} \frac{cut(C_j, \bar{C}_j)}{\mathcal{V}'_j} = \sum_{j=1}^{k} \frac{1}{\mathcal{V}'_j} \sum_{i \in C_j} \sum_{\ell \notin C_j} s_{i\ell} \tag{8c}$$

where

$$\mathcal{V}'_j = \sum_{i \in C_j} (d_{ii} + 1) = \mathcal{V}_j + |C_j|\,. \tag{9}$$

The last formula allows us to consider the NRCut clustering criterion as a mixture of NCut and RCut.

Note the symmetry between the k-means clustering criterion and the cutting criterion. In (4), the averaged difference (measured by the Euclidean distance) between the members of the j-th cluster is minimized, while in (8), the averaged similarity between neighboring nodes assigned to different clusters is minimized, respectively.

4. Our Approach

Our approach to explanation differs from that presented, e.g., by [18] as their methodology is (1) targetting explanation for classification while we aim at explanation of clustering results, (2) the classification mechanism they use is a black-box itself, while we aim at a methodology linking the clustering result cleanly to the textual content of documents, without referring to mysterious coefficients.

4.1. Preparing BERT Embedding for Explanation

As recalled in section 2, one can get an embedding of an entire sentence as well as embeddings of each word occurrence in the sentence in multiple ways. Let us try to get a kind of uniform representation for them. Assume we have a collection $\mathcal{D}$ of documents $\delta_1, \dots, \delta_n$ and a dictionary $\mathcal{T}$ consisting of words $\mathfrak{w}_1, \dots, \mathfrak{w}_m$. A word $\mathfrak{w}_t$ occurs in document δ_i $o(t,i)$ times. Let us define a function $\mathcal{E}(\delta_i)$ returning the BERT embedding in the space $\mathbb{R}^d$ of the document δ_i and a function $\mathcal{E}(\mathfrak{w}_t, p, \delta_i)$ returning the BERT embedding of the p^{th} occurrence in the space $\mathbb{R}^w$ of the word $\mathfrak{w}_t$ in the document δ_i [11]. The dimensions d and w are assumed to be identical.

Furthermore, as we want to compute cosine similarities, assume that $\mathcal{E}^*(\delta_i) = \mathcal{E}(\delta_i)/||\mathcal{E}(\delta_i)||$, and $\mathcal{E}^*(\mathfrak{w}_t, p, \delta_i) = \mathcal{E}(\mathfrak{w}_t, p, \delta_i)/||\mathcal{E}(\mathfrak{w}_t, p, \delta_i)||$.

Frequently, the embedding extraction technologies induce that $\mathcal{E}(\delta_i)$ is a linear combination of all occurrences $\mathfrak{w}_{j,p}$, but it does not need to be so [15].

However, the dimensionality d is higher than the maximum number of word occurrences in a document, we can assume that there exists a set of non-negative coefficients f for each document δ_i such that $|| \sum_{\mathfrak{w}_t \in \delta_i} \sum_{p=1}^{o(t,i)} f_{i,j,p} \mathcal{E}^*(\mathfrak{w}_t, p, \delta_i) - \mathcal{E}^*(\delta_i)||$ is minimized (constrained minimization).
In other words

$$\mathcal{E}^*(\delta_i) \approx \sum_{\mathfrak{w}_t \in \delta_i} \sum_{p=1}^{o(t,i)} f_{i,j,p} \mathcal{E}^*(\mathfrak{w}_t, p, \delta_i) \,. \tag{10}$$

Let us define the importance (or contribution) of a word for the document in such a way that the sum of the embeddings of all occurring words is equal to 1 (approximately), while closeness of a word embedding vector to the document embedding vector is reflected by higher values.

$$importance(\mathfrak{w}_t, \delta_i) = \sum_{p=1}^{o(t,i)} f_{i,j,p} \mathcal{E}^*(\mathfrak{w}_t, p, \delta_i)^T \mathcal{E}^*(\delta_i) \,. \tag{11}$$

Clearly, if the document embedding is a linear combination of token embeddings, then $\sum_{\mathfrak{w}_t \in \delta_i} importance(\mathfrak{w}_t, \delta_i) = 1$

Let us cluster the documents in this embedding using k-means. That is, we minimize the loss function

$$Q^{[BERT]}(\Gamma) = \sum_{j=1}^{k} \sum_{i \in C_j} ||\mathcal{E}^*(\delta_i) - \mu(C_j)||^2 \tag{12}$$

where $\Gamma = \{C_1, \dots, C_k\}$ is a partition of the set of documents.

Each cluster C_j will have a cluster center (or prototype) $\mu(C_j)$ such that

$$\mu(C_j) = \frac{1}{|C_j|} \sum_{\delta_i \in C_j} \mathcal{E}^*(\delta_i) \,. \tag{13}$$

By analogy, the importance of a word for the cluster may be defined as

$$importance(\mathfrak{w}_t, C) = \\ = \mu(C)^T \left(\frac{1}{|C|} \sum_{\delta_i \in C} \sum_{p=1}^{o(t,i)} f_{i,j,p} \mathcal{E}^*(\mathfrak{w}_t, p, \delta_i) \right) . \tag{14}$$

By sorting the words by their decreasing importance, we get the knowledge which word explains best cluster membership.

Note, that

$$importance(\mathfrak{w}_t, C) = \frac{1}{|C|} \sum_{\delta_i \in C} importance(\mathfrak{w}_t, \delta_i) \,. \tag{15}$$

4.2. Weighted Clustering in BERT Embedding

Consider now a modified vector in the BERT embedding. Let $\omega_i = d_{ii}$.

$$\mathcal{U}^*(\delta_i) = \left(\frac{\mathcal{E}^*(\delta_i)}{\omega_i}, \mathbf{g_i} \right) \tag{16}$$

where $\mathbf{g_i}$ is a vector of dimension n, equal to zero everywhere except the ith element, $g_{ii} = \frac{\sqrt{\omega_i - 1}}{\omega_i}$. With this notation, let us compute the squared distance between two documents: (for different i, ℓ)

$$||\mathcal{U}^*(\delta_i) - \mathcal{U}^*(\delta_\ell)||^2 = $$
$$= ||\mathcal{U}^*(\delta_i)||^2 + ||\mathcal{U}^*(\delta_\ell)||^2 - 2\mathcal{U}^*(\delta_\ell)^T \mathcal{U}^*(\delta_i) \tag{17}$$
$$= \frac{\omega_i - 1 + 1}{\omega_i^2} + \frac{\omega_\ell - 1 + 1}{\omega_\ell^2} - 2\frac{s_{i\ell}}{\omega_i \omega_\ell} \tag{18}$$
$$= \frac{1}{\omega_i} + \frac{1}{\omega_\ell} - 2\frac{s_{i\ell}}{\omega_i \omega_\ell} \,. \tag{19}$$

The dissimilarity is greater when the vectors are longer, but smaller if the dot product is bigger. Under this assumption, let us perform weighted k-means clustering (for weighted kernel-k-means see, e.g., [22]). That is, we minimize the loss function

$$Q^{[\omega BERT]}(\Gamma; \omega) = \sum_{j=1}^{k} \sum_{i \in C_j} \omega_i ||\mathcal{U}^*(\delta_i) - \mu_\omega(C_j)||^2 \tag{20}$$

where $\Gamma = \{C_1, \ldots, C_k\}$ is a partition of the set of documents. Each cluster C_j will have a cluster center (or prototype) $\mu_\omega(C_j)$ such that

$$\mu_\omega(C_j) = \frac{1}{\sum_{\delta_i \in C_j} \omega_i} \sum_{\delta_i \in C_j} \omega_i \mathcal{U}^*(\delta_i) \tag{21}$$

$$Q^{[\omega BERT]}(\Gamma; \omega) =$$

$$= \sum_{j=1}^{k} \frac{1}{2\mathcal{V}_j} \sum_{i \in C_j} \sum_{\substack{\ell \in C_j \\ \ell \neq i}} \omega_i \omega_\ell \|\mathcal{U}^*(\delta_i) - \mathcal{U}^*(\delta_\ell)\|^2 \tag{22}$$

$$= \sum_{j=1}^{k} \frac{1}{2\mathcal{V}_j} \sum_{i \in C_j} \sum_{\substack{\ell \in C_j \\ \ell \neq i}} \omega_i \omega_\ell \left(\frac{1}{\omega_i} + \frac{1}{\omega_\ell} - 2\frac{s_{i\ell}}{\omega_i \omega_\ell} \right) \tag{23}$$

$$= \sum_{j=1}^{k} \frac{1}{2\mathcal{V}_j} \sum_{i \in C_j} \sum_{\substack{\ell \in C_j \\ \ell \neq i}} (\omega_i + \omega_\ell - 2s_{i\ell}) \ . \tag{24}$$

Note, that the $\mathcal{U}^*(\delta_i)$ vectors are of higher dimension. There is no need to use them in practice during the clustering process. One uses them only "mentally" for the sake of explanation. As proven in [23], clusters resulting from clustering the $\mathcal{U}^*(\delta_i)$ vectors are the same as clusters resulting from clustering using GSC based on normalized Laplacians.

As for explanation purposes note, that

$$\mu_\omega(C_j) = \frac{1}{\sum_{\delta_i \in C_j} \omega_i} \sum_{\delta_i \in C_j} \left(\mathcal{E}^*(\delta_i), \omega_i \mathbf{g_i} \right) \tag{25}$$

$$\mu_\omega(C_j) = \frac{1}{\sum_{\delta_i \in C_j} \omega_i} \Big(\sum_{\delta_i \in C_j} \mathcal{E}^*(\delta_i), \sum_{\delta_i \in C_j} \omega_i \mathbf{g_i} \Big) \tag{26}$$

$$\mu_\omega(C_j) = \frac{1}{\sum_{\delta_i \in C_j} \omega_i} \Big(|C_j| \mu(C_j), \sum_{\delta_i \in C_j} \omega_i \mathbf{g_i} \Big) . \tag{27}$$

which means that the word importance can be computed in analogy to the unweighted case, but with the distinction that potentially the clusters are different due to weighted clustering.

The weighted importance can be defined as

$$importance_\omega(\mathfrak{w}_t, C) = \tag{28}$$

$$= \mu_\omega(C_j)^T \left(\frac{1}{|C|} \sum_{\delta_i \in C} \sum_{p=1}^{o(t,i)} f_{i,j,p} \left(\mathcal{E}^*(\delta_i), \omega_i \mathbf{g_i} \right) \right)$$

$$= \mu_\omega(C_j)^T \left(\frac{1}{|C|} \sum_{\delta_i \in C} \left(\mathcal{E}^*(\delta_i), \omega_i \mathbf{g_i} \right) \sum_{p=1}^{o(t,i)} f_{i,j,p} \right) .$$

4.3. A Proposal of Double-centered "Normalized" Document Similarity Matrix Based Embedding (For Use With Weighted k-Means)

In [23], it has been suggested to use the $\mathcal{A}$ matrix of the following form. $\mathfrak{E}$ be a matrix of the following form

$$\mathfrak{E} = \mathbf{1}\mathbf{1}^T - I \ . \tag{29}$$

Then define

$$\mathcal{A} = D^{-1}(\mathfrak{E}D + D\mathfrak{E} - 2S)D^{-1} \ . \tag{30}$$

with D, S being defined as previously. Let $\mathcal{M}$ be the matrix of the form:

$$\mathcal{M} = -\frac{1}{2}(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)\mathcal{A}(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T) \ . \tag{31}$$

Note, that $\mathbf{1}$ is an eigenvector of $\mathcal{M}$, with the corresponding eigenvalue equal to 0. All the other eigenvectors must be orthogonal to it as $\mathcal{M}$ is real and symmetric, so for any other eigenvector $\mathbf{v}$ of $\mathcal{M}$ we have: $\mathbf{1}^T\mathbf{v} = 0$.

Let Λ be the diagonal matrix of eigenvalues of $\mathcal{M}$, and V the matrix where columns are corresponding (unit length) eigenvectors of $\mathcal{M}$. Then $\mathcal{M} = V\Lambda V^T$. Let $\zeta_i = \Lambda^{1/2}V_i^T$, where V_i stands for the i-th row of V. Let ζ_i, ζ_ℓ be the embeddings of the documents i, ℓ, resp. This embedding shall be called *$\mathcal{M}$-embedding*. Then

$$\|\zeta_i - \zeta_\ell\|^2 = \mathcal{A}_{i\ell} = (d_{ii} + d_{\ell\ell} - 2s_{i\ell})/(d_{ii}d_{\ell\ell}) \tag{32}$$

for $i \neq \ell$, and zero otherwise. Let us now discuss performing weighted k-means clustering on the vectors ζ_i with weights amounting to d_{ii} respectively.

Let us use the following weighting of documents: $\omega_i = d_{ii}$. Clustering via weighted k-means with weights ω_i in the $\mathcal{M}$ embedding will optimize the following criterion

$$Q^{[\mathcal{M}based]}(\Gamma; \omega) = \sum_{j=1}^{k} \sum_{i \in C_j} \omega_i \|\zeta_i - \mu_\omega(C_j)\|^2 \tag{33}$$

whereby

$$\mu_\omega(C_j) = \frac{\sum_{\in C_j} \omega_i \zeta_i}{\sum_{\in C_j} \omega_i} = \frac{1}{\mathcal{V}_j} \sum_{\in C_j} \omega_i \zeta_i \ . \tag{34}$$

In [23] it has been proven that

$$Q^{[\mathcal{M}based]}(\Gamma; \omega) = n - 2k + NCut(\Gamma) \tag{35}$$

Recall, that $NCut(\Gamma)$ was defined in equation (8c). Since $n - 2k$ is a constant, minimizing one criterion minimizes the other. As N-based Clustering (clustering using the normalized Laplacian $\mathfrak{L}$) has the same target as NCut clustering, they are equivalent (see sec. 3).

In [23], proof can be found that

$$Q^{[\mathcal{M}based]}(\Gamma; \omega) = \sum_{j=1}^{k} \frac{1}{2\mathcal{V}_j} \sum_{i \in C_j} \sum_{\substack{\ell \in C_j \\ \ell \neq i}} (d_{ii} + d_{\ell\ell} - 2s_{i\ell}) \ . \tag{36}$$

It is easily seen that it is identical with the equation (24) showing equivalence of $Q^{[\mathcal{M}based]}(\Gamma; \omega)$ with $Q^{[\omega BERT]}(\Gamma; \omega)$.

This completes the explanation bridge: you can cluster with normalized Laplacian based GSC and explain with weighted BERT explanation method. The advantage is that the clustering is performed in a much lower dimensional space (k dimensions when seeking k clusters), but without the disadvantage of basic GSC of non-explainability. Explanation is reached in the BERT space.

4.4. Algorithmic Description of Clustering with Explanation Under BERT Embedding

The outlined methodology can be summarized as follows:

1) Embed the document collection into BERT.
2) For each document $\mathcal{E}(\delta_i)$ compute the normalized embedding vector $\mathcal{E}^*(\delta_i)$.
3) Compute cosine similarity between documents i, ℓ as cosine between the embedding vectors of documents $s_{i\ell}\mathcal{E}^*(\delta_i)^T\mathcal{E}^*(\delta_\ell)$ forming the similarity matrix S.
4) For the similarity matrix S, apply normalized Laplacian based clustering, eq. (4).
5) For each token $\mathcal{E}(\mathfrak{w}_t, p, \delta_i)$ of the document compute the normalized embedding vector $\mathcal{E}^*(\mathfrak{w}_t, p, \delta_i)$.
6) For each document, compute the linear approximation of the normalized document embedding vector based on normalized token embedding vectors eq. (10).
7) For each cluster, with center eq. (21), calculate the importance of words associated with the cluster eq. (28).
8) Take n most important words as the cluster explanation for each cluster.

5. Experiments

In [23], it has been shown that explainability can be achieved when performing GSC for similarities obtained in the Term Vector Space. Therefore, the question can be raised: What is gained when considering the BERT embedding? Experiments have been performed showing the improved clustering performance of the latter.

For this purpose, we studied the effectiveness of BERT-based clustering versus Glove and traditional TVS embeddings. The sBERT and BERT embedding models were downloaded automatically (in the background) by libraries used for embedding of textual documents, whereby model names needed to be provided. The models are available, for example, in the huggingface repository (`https://huggingface.co/models`). Alternatively, manual download is also possible. Python library used for BERT embeddings was `transformers`. Models used in experiments were: `bert-base-uncased`, `vinai/bertweet-base`, `distilbert-base-uncased`, `cardiffnlp/twitter-roberta-base`. Python library used for sBERT embeddings is called `sentence_transformers`. The names of sBERT and BERT embeddings visible in the tables 1 - 5, consist of two and three parts, respectively, separated with and #. The first part (sBERT, BERT) indicates the type of embedding. The second part is the common name of the respective embedding model. The last part, for BERT embeddings, following #, indicates the version of embedding extraction. Two different document embedding extractors were considered: those marked with #[CLS] mean embeddings taken from the [CLS] token, while #T_AVG indicate that the document embedding was taken as the plain average of (the other) token embeddings.

GloVe-based embeddings are the ones trained on Twitter data (GloVe@twitter) and trained on Wikipedia data (Glove@wiki), downloaded from the pages indicated in [10]. The embeddings with traditional TVS (tf, tfidf) are based on the Python `scikit-learn` library.

The clustering method over all the embeddings was N-based clustering. sBERT embeddings differ from BERT embeddings in that for BERT embeddings, we get both document embedding and embeddings of individual tokens. This enables explanations to be derived. However, sBERT provides document embedding only. In this study of clustering efficiency, we compare sBERT and BERT embeddings to see whether or not sBERT-based clusterings perform significantly better than BERT embeddings. If they do not perform significantly better, then we can use BERT embeddings for clustering and can be satisfied with the explanation methodology provided in this paper. If sBERT were significantly better, separate explanation methods need to be sought for sBERT.

The clustering experiments were performed with popular Python libraries: `numpy` [24], `scipy` [25], `scikit-learn` [26] and `soyclustering` [27] which is an implementation of spherical k-means [28]. In particular, we used `SpectralClustering` class from scikit-learn with the `affinity` parameter set to: `precomputed` (affinity from similarity matrix) as a representative of the N-embedding based clustering.

The Hungarian method [29] is applied to match hashtags and clusters, aiming to achieve the best agreement (lowest error rate).

We made the comparison based on 5 datasets drawn from Twitter. Each dataset consisted of documents related to one of five hashtags only. Below, a short characterization of each dataset is given.

- DataSet 0: 8050 documents with 23855 distinct terms. It consists of 5 hashtags: mentalhealth (1003 docs), brexit (1607), ukraine (1687), covid_19 (1696), writingcommunity (2057).
- DataSet 1: 8159 documents with 21188 distinct terms. It consists of 5 hashtags: bbcqt (1025 docs), sidnaaz (1153), lufc (1470), 100daysofcode (1718), tejran (2793).
- Dataset 2: 8256 documents with 20217 distinct terms, and 5 hashtags: r4today (1036 docs), rhobh (1298), bb22 (1622), blm (1849), cdnpoli (2451).
- Dataset 3: 8218 documents with 21061 distinct terms, and 5 hashtags: smackdown (1063 docs), rhop (1154), btc (1487), maga (2079), nufc (2435).
- Dataset 4: 8440 docs with 26939 distinct terms, and 5 hashtags: dnd (1031 docs), robostopia (1323), browns (1493), aewdynamite (1821), 2 (2772).

To measure the quality of clustering, we compared the clustering results with the human-made clustering in terms of the hashtags assigned to documents. Out

Table 1. N-based clustering of Dataset 0 using various embeddings (vectorizers)

Vectorizer	F1-avg	F1-stdev
CountVectorizer	0.255152	0.000049
TfVectorizer	0.255146	0.000102
TfidfVectorizer	0.396431	0.000360
GloVe@wiki	0.363963	0.000401
GloVe@twitter	0.355927	0.001543
sBERT@all-MiniLM-L6-v2	0.974979	0.000092
sBERT@all-distilroberta-v1	0.951857	0.000382
sBERT@multi-qa-mpnet-base-dot-v1	0.942263	0.000090
BERT@bert-base-uncased#[CLS]	0.472723	0.000262
BERT@vinai/bertweet-base#[CLS]	0.675733	0.000439
BERT@distilbert-base-uncased#[CLS]	0.592854	0.000481
BERT@cardiffnlp/twitter-roberta-base#[CLS]	0.831508	0.000091
BERT@bert-base-uncased#T_AVG	0.602534	0.000423
BERT@vinai/bertweet-base#T_AVG	0.618835	0.000151
BERT@distilbert-base-uncased#T_AVG	0.558688	0.000899
BERT@cardiffnlp/twitter-roberta-base#T_AVG	0.476953	0.002215

Table 2. N-based clustering of Dataset 1 using various embeddings (vectorizers)

Vectorizer	F1-avg	F1-stdev
CountVectorizer	0.367469	0.000429
TfVectorizer	0.367626	0.000624
TfidfVectorizer	0.497295	0.001748
GloVe@wiki	0.463999	0.000942
GloVe@twitter	0.594311	0.001918
sBERT@multi-qa-mpnet-base-dot-v1	0.906746	0.000166
BERT@bert-base-uncased#[CLS]	0.475842	0.000682
BERT@vinai/bertweet-base#[CLS]	0.573888	0.000470
BERT@distilbert-base-uncased#[CLS]	0.654812	0.000458
BERT@cardiffnlp/twitter-roberta-base#[CLS]	0.799531	0.000085
BERT@bert-base-uncased#T_AVG	0.609975	0.001175
BERT@vinai/bertweet-base#T_AVG	0.683997	0.000223
BERT@distilbert-base-uncased#T_AVG	0.694325	0.000253
BERT@cardiffnlp/twitter-roberta-base#T_AVG	0.727976	0.000198

of the multitude of possibilities (e.g., [30] or [31]), as a quality measure for clustering, we took the popular external quality F1 measure as it reflects both precision and recall. Based on the Hungarian method, we associate each cluster with a single hashtag, calculate F1 for each hashtag against the rest. Then compute the average for all hashtags.

We had to refrain from evaluation of the explanations limiting ourselves to visual inspection (see an example below) as no appropriate reference data sets for the tweets we considered are available. We are currently developing an evaluation method that does not require human intervention and is therefore objective. This paper establishes only the theoretical basis for the explanation process itself. We refrain from adding unnecessary complexity to this paper. There exist multiple works [32–36] and surveys on assessing the quality of explanations like [37–40]. Most quality evaluation methods require either direct interaction with human user or some predefined sets with ”ground truth” explanations. We chose in this paper a different pathway and provided an analytical justification why the explanations are trustworthy.

Tables 1 - 5 present clustering results for datasets 0 - 4, respectively. The F1-score presented therein is an average over 30 runs. Note, that in some cases, GSC clustering (as based on normalized Laplacian) failed due to negative similarities. Most frequently, sBERT was affected, but also in some cases GloVe and BERT embedding.

In all cases except for dataset 4, sBERT embedding methods yield the best clustering results. They are, however, not the best option for explanation purposes as an approximation equation. (10) constitutes a poor approximation of document embeddings via token embeddings.

BERT embeddings are not as good as sBERT embeddings. However, if we compare #[CLS] embeddings with #T_AVG in the case of BERT methods, they do not differ very much. Hence, the usage of the averaging method for document embedding is well justified and paves the way for the application of the eq. (10) approximation, and hence the proposed explanation method is justifiable. And, as already mentioned, sBERTs are more problematic than BERTs due to failures caused by frequent negative similarities.

BERT methods are generally better than the TVS and GloVe-based embeddings, but not in all cases (as stated also in, e.g., [41]).

Explanations of clusters under BERT embeddings can be essentially used in a meaningful way only when the embedding of the document is the average over token embeddings (#T_AVG variants). One faces a similar problem as in other settings, that is, the stopwords play a non-negligible role. See an example of a cluster description with top 50 tokens: *the to of # mentalhealth and a is in covid writingcommunity for you are this that we @ _ i on 19 your be all - with it have not & people health our an can from ukraine mental will need those at about my their so who by as.*

Table 3. N-based clustering of Dataset 2 using various embeddings (vectorizers)

Vectorizer	F1-avg	F1-stdev
CountVectorizer	0.277486	0.000189
TfVectorizer	0.277534	0.000169
TfidfVectorizer	0.575343	0.000483
GloVe@twitter	0.385300	0.000959
sBERT@all-MiniLM-L12-v2	0.834372	0.000242
sBERT@multi-qa-distilbert-cos-v1	0.820115	0.000252
sBERT@multi-qa-mpnet-base-dot-v1	0.854759	0.000197
BERT@bert-base-uncased#[CLS]	0.488475	0.000369
BERT@vinai/bertweet-base#[CLS]	0.780850	0.000328
BERT@distilbert-base-uncased#[CLS]	0.464572	0.000172
BERT@cardiffnlp/twitter-roberta-base#[CLS]	0.632636	0.000423
BERT@bert-base-uncased#T_AVG	0.478280	0.000458
BERT@vinai/bertweet-base#T_AVG	0.540016	0.003658
BERT@distilbert-base-uncased#T_AVG	0.444183	0.000424
BERT@cardiffnlp/twitter-roberta-base#T_AVG	0.551978	0.000632

Table 4. N-based clustering of Dataset 3 using various embeddings (vectorizers)

Vectorizer	F1-avg	F1-stdev
CountVectorizer	0.231842	0.001354
TfVectorizer	0.231546	0.001468
TfidfVectorizer	0.320499	0.000485
GloVe@twitter	0.444521	0.000620
sBERT@all-MiniLM-L6-v2	0.985108	0.000009
sBERT@all-MiniLM-L12-v2	0.986921	0.000000
sBERT@multi-qa-MiniLM-L6-cos-v1	0.960467	0.000130
sBERT@multi-qa-mpnet-base-dot-v1	0.978597	0.000071
BERT@bert-base-uncased#[CLS]	0.537486	0.000581
BERT@vinai/bertweet-base#[CLS]	0.641715	0.000124
BERT@distilbert-base-uncased#[CLS]	0.633663	0.000141
BERT@cardiffnlp/twitter-roberta-base#[CLS]	0.935586	0.000230
BERT@bert-base-uncased#T_AVG	0.652909	0.000174
BERT@vinai/bertweet-base#T_AVG	0.713428	0.000090
BERT@distilbert-base-uncased#T_AVG	0.611432	0.000196
BERT@cardiffnlp/twitter-roberta-base#T_AVG	0.677452	0.000217

Same cluster described after removing the stopwords looks like this: *mentalhealth covid writingcommunity people health ukraine mental support time pandemic love care social day distancing safe life crisis issues feel lives remember complacency current depression medical zelenskyyua future llinzigray wip feeling decisionproject government based school banffacademybxa virus children talk psychosocial public equipment retweeting uncivilized power flyfofaaviation foodsafetygov money brexit writing*, which seems to be more topical.

Although removal of stopwords improves the quality of the description, one issue remains. BERT like models create embeddings based on the entire document and there is no obvious justification for discarding stopwords from the text when knowing the specific way how the models are trained (transformer method). The effects of removing stopwords under these settings would require an in-depth investigation. Whereas embeddings like GloVe, TVS that are performed on a word-by-word basis are free from this ambiguity – we can simply ignore the stopwords under embeddings.

6. Conclusions

In this paper, we studied the problems behind explainability of the results of Graph Spectral Clustering (GSC) methods applied under diverse types of document embeddings, belonging to the TVS group, GloVe group and BERT group.

The essential problem with explainability under GSC is that the GSC embeddings of the results of clustering have nothing to do with the document content. Therefore, [23] suggested a multistage explanation process which is applicable to TVS embedding types. In [42], an extension was discussed to GloVe type embeddings which is more complex than that for TVSembeddings.

The mentioned papers constitute a kind of breakthrough in the domain of GSA. GSA was previously considered as a black-box method, while the mentioned articles make it explainable. Hereby, there is a grading of complexity on the side of the NL embeddings. TVS represents a relatively simple case where the weight of explaining word is directly the TVS coordinate. GloVe embeddings are more complicated because of intersection of word vectors in this space. Though this is still a simpler case than BERT, as each word has a fixed embedding. BERT family has a different embedding of each word not only in different documents, but also within the same document. The embedding of the entire document may not correspond to a linear combination of word tokens.

As demonstrated in this paper, a still more complex extension of the explanation concept is also possible for BERT embeddings.

Table 5. N-based clustering of Dataset 4 using various embeddings (vectorizers)

Vectorizer	F1-avg	F1-stdev
CountVectorizer	0.238978	0.000000
TfVectorizer	0.238978	0.000000
TfidfVectorizer	0.239120	0.000000
GloVe@wiki	0.292245	0.000981
GloVe@twitter	0.327367	0.059270
sBERT@multi-qa-mpnet-base-dot-v1	0.670961	0.000124
BERT@bert-base-uncased#[CLS]	0.532433	0.000220
BERT@vinai/bertweet-base#[CLS]	0.450124	0.000068
BERT@distilbert-base-uncased#[CLS]	0.644404	0.000593
BERT@cardiffnlp/twitter-roberta-base#[CLS]	0.769169	0.000094
BERT@bert-base-uncased#T_AVG	0.614746	0.000177
BERT@vinai/bertweet-base#T_AVG	0.592477	0.000221
BERT@distilbert-base-uncased#T_AVG	0.663272	0.000055
BERT@cardiffnlp/twitter-roberta-base#T_AVG	0.672432	0.000529

We faced two challenges: (1) deriving theoretical formulas of extracting word importance under the assumption of linear combination of token embeddings (2) experimental checking if usage of linear word token combination deteriorates performance compared to the document embedding (CLS token). Managing these two points makes the significant difference to the previous research and allows to say: clustering with GSA under BERT is also explainable.

The question was also raised as to whether or not it is worth the extra work to use the BERT-based embeddings. The accuracy of clustering for embeddings of these different classes of doc embeddings were studied. BERT embeddings appear to yield best results.

However, an issue was observed that needs a future investigations. Similarly to GloVe embeddings, it may happen that BERT embeddings produce negative cosine similarity between documents. An investigation similar to that described in [42] is needed to ensure completeness of applicability of explnations under BERT embeddings.

Also, a deeper investigation is needed with respect to the role of stopwords in BERT, both on their impact of clustering quality and explanation quality. The research question is: shall we remove stopwords (1) before starting the process of BERT embedding, (2) after it but before computing the embedding of the document based on tokens for clustering or (3) after the clustering but before explaining.

Note, that a number of researchers insist that not the explainability but rather the interpretability of ML results is more important, see [43, 44], but we see it differently. Interpretability is one side of ML usage pointing at the possibility to ”make money” out of the ML results. However, the other side of the results is their trustfulness – investment risk. One has to understand how the results came about. The results must be hence explainable.

We restricted ourselves to the BERT-type document embedding models, although new ones are constantly being developed, such as those listed on the leaderboard at https://huggingface.co/spaces/mteb/leaderboard. We chose, however, BERT due to its availability and broad usage, and due to its open source nature and relatively low computational demands. We look at the path TVS-GloVe-BERT as a pathway towards development of more general explanation methods for the steadily evolving LLM models.

While there are multiple ways of evaluating clusters, we concentrate on comparison with ”intrinsic” clustering that is hashtags. Based on Hungarian method, we associate each cluster with a single hashtag, calculate F1 for each hashtag against the rest. Then compute the average for all hashtags.

Notes

[1] Semantic similarity is nevertheless an ongoing subject of research; see, e.g., [45].

[2] For a list of some BERT model variants available from Huggingface see https://www.kaggle.com/datasets/sauravmaheshkar/huggingface-bert-variants. For a list of other transformer models, see, e.g., https://github.com/abacaj/awesome-transformers.

[3] Other Laplacians are also used, e.g., the random walk Laplacian $\mathbb{L}$ of a graph, defined as

$$\mathbb{L} = LD^{-1} = I - SD^{-1} \quad (37)$$

Other Laplacians were also studied [21].

AUTHORS

Mieczysław A. Kłopotek* – Instytut Podstaw Informatyki Polskiej Akademii Nauk, ul. Jana Kazimierza 5, 01-248 Warszawa, Poland, e-mail: klopotek@ipipan.waw.pl, https://home.ipipan.waw.pl/m.klopotek/.

Sławomir T. Wierzchoń – Instytut Podstaw Informatyki Polskiej Akademii Nauk, ul. Jana Kazimierza 5, 01-248 Warszawa, Poland, e-mail: stw@ipipan.waw.pl, https://home.ipipan.waw.pl/s.wierzchon/.

Bartłomiej Starosta – Instytut Podstaw Informatyki Polskiej Akademii Nauk, ul. Jana Kazimierza 5, 01-248 Warszawa, Poland, e-mail: barstar@ipipan.waw.pl, https://home.ipipan.waw.pl/b.starosta/ .

Piotr Borkowski – Instytut Podstaw Informatyki Polskiej Akademii Nauk, ul. Jana Kazimierza 5, 01-248 Warszawa, Poland, e-mail: p.borkowski@ipipan.waw.pl, https://home.ipipan.waw.pl/p.borkowski/.

Dariusz Czerski – Instytut Podstaw Informatyki Polskiej Akademii Nauk, ul. Jana Kazimierza 5, 01-248 Warszawa, Poland, e-mail: dcz@ipipan.waw.pl, https://home.ipipan.waw.pl/d.czerski/.

*Corresponding author

References

[1] S. A. P. Murray, *The Library: An Illustrated History*, Skyhorse Pub.: New York, NY, 2009.

[2] S. Lloyd, "Least squares quantization in PCM", *IEEE Transactions on Information Theory*, vol. 28, no. 2, 1982, 129–137, 10.1109/TIT.1982.1056489.

[3] U. Luxburg von, "A tutorial on spectral clustering", *Statistics and Computing*, vol. 17, no. 4, 2007, 395–416, https://doi.org/10.1007/s11222-007-9033-z.

[4] J. Wang and Y. Dong, "Measurement of text similarity: A survey", *Information*, vol. 11, no. 9, 2020, 10.3390/info11090421.

[5] A. Ittoo, L. M. Nguyen, and A. van den Bosch, "Text analytics in industry", *Computers in Industry*, vol. 78, no. C, 2016, 96–107, 10.1016/j.compind.2015.12.001.

[6] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing", *Communications of the ACM*, vol. 18, no. 11, 1975, 613–620, 10.1145/361219.361220.

[7] A. P. Sunita Bisht, "Document clustering: A review", *International Journal of Computer Applications*, vol. 73, no. 11, 2013, 26–33, 10.5120/12787-0024.

[8] X. Rong. "word2vec parameter learning explained", 2014. arXiv:1411.2738 [cs.CL].

[9] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation". In: *Proceedings of the 1st Workshop on Representation Learning for NLP*, Berlin, Germany, 2016, 78–86, https://doi.org/10.18653/v1/W16-1609.

[10] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation". In: A. Moschitti, B. Pang, and W. Daelemans, eds., *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, 1532–1543, 10.3115/v1/D14-1162.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding", *Computing Research Repository*, vol. abs/1810.04805, 2018, https://doi.org/10.48550/arXiv.1810.04805.

[12] E. Sezerer and S. Tekir, "A survey on neural word embeddings", *Computing Research Repository*, vol. abs/2110.01804, 2021.

[13] Y. Li and T. Yang. "Word embedding for understanding natural language: A survey". In: S. Srinivasan, ed., *Guide to Big Data Applications*, 83–104. Springer International Publishing, Cham, 2018.

[14] L. da Costa, I. Oliveira, and R. Fileto, "Text classification using embeddings: a survey", *Knowledge and Information Systems*, vol. 65, 2023, 2761–2803, https://doi.org/10.1007/s10115-023-01856-z.

[15] L. Stankevičius and M. Lukoševičius, "Extracting sentence embeddings from pretrained transformer models", *Computing Research Repository*, vol. abs/2408.08073, 2024, https://doi.org/10.3390/app14198887.

[16] S. Talebi, E. Tong, A. Li, and et al., "Exploring the performance and explainability of fine-tuned bert models for neuroradiology protocol assignment", *BMC Medical Informatics and Decision Making*, vol. 24, no. 40, 2024, 10.1186/s12911-024-02444-z.

[17] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, and O. Reblitz-Richardson, "Captum: A unified and generic model interpretability library for pytorch", *Computing Research Repository*, vol. abs/2009.07896, 2020, https://doi.org/10.48550/arXiv.2009.07896.

[18] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks". In: *Proc. 34th International Conference on Machine Learning*, vol. 70, 2017, 3319–28.

[19] N. O. Tan, J. Bensemann, D. Benavides-Prado, Y. Chen, M. Gahegan, L. Lee, A. Y. Peng, P. Riddle, and M. Witbrock, "An explainability analysis of a sentiment prediction task using a transformer-based attention filter". In: *Proceedings of the Ninth Annual Conference on Advances in Cognitive Systems*, 2021.

[20] Z. Xu and Y. Ke, "Effective and efficient spectral clustering on text and link data". In: *CIKM '16: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, 357–366, https://doi.org/10.1145/2983323.2983708.

[21] S. Wierzchoń and M. Kłopotek, *Modern Clustering Algorithms*, volume 34 of *Studies in Big Data*, Springer Verlag, 2018, 421.

[22] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: Spectral clustering and normalized cuts". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2004, 551–556, 10.1145/1014052.1014118.

[23] B. Starosta, M. A. Kłopotek, S. T. Wierzchoń, D. Czerski, M. Sydow, and P. Borkowski, "Explainable graph spectral clustering of text documents", *PLoS One*, vol. 20(2):e0313238, 2025, 10.1371/journal.pone.0313238.

[24] C. R. Harris, K. J. Millman, S. J. van der Walt, et al., "Array programming with NumPy", *Nature*, vol. 585, 2020, 357–362, 10.1038/s41586-020-2649-2.

[25] P. Virtanen, R. Gommers, T. E. Oliphant, et al., "SciPy 1.0: Fundamental algorithms for scientific computing in python", *Nature Methods*, vol. 17, 2020, 261–272, 10.1038/s41592-019-0686-2.

[26] L. Buitinck, G. Louppe, M. Blondel, et al., "API design for machine learning software: experiences from the scikit-learn project", *Computing Research Repository*, vol. abs/1309.0238, 2013, https://doi.org/10.48550/arXiv.1309.0238.

[27] H. Kim and H. K. Kim. "clustering4docs github repository", 2020. https://pypi.org/project/soy clustering/.

[28] H. Kim, H. K. Kim, and S. Cho, "Improving spherical k-means for document clustering: Fast initialization, sparse centroid projection, and efficient cluster labeling", *Expert Systems with Applications*, vol. 150, 2020, 113288, https://doi.org/10.1016/j.eswa.2020.113288.

[29] H. W. Kuhn, "The hungarian method for the assignment problem", *Naval Research Logistics Quarterly*, vol. 2, 1955, 83–97.

[30] D. Pfitzner, R. Leibbrandt, and D. Powers, "Characterization and evaluation of similarity measures for pairs of clusterings", *Knowledge and Information Systems*, vol. 19, no. 3, 2009, 361–394, https://doi.org/10.1007/s10115-008-0150-6.

[31] E. Achtert, S. Goldhofer, H.-P. Kriegel, E. Schubert, and A. Zimek, "Evaluation of clusterings – metrics and visual support". In: *2012 IEEE 28th International Conference on Data Engineering*, 2012, 1285–1288, 10.1109/ICDE.2012.128.

[32] A. Balagopalan, H. Zhang, K. Hamidieh, T. Hartvigsen, F. Rudzicz, and M. Ghassemi, "The road to explainability is paved with bias: Measuring the fairness of explanations". In: *2022 ACM Conference on Fairness Accountability and Transparency*, 2022, 1194–1206, 10.1145/3531146.3533179.

[33] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman, "Measures for explainable AI: Explanation goodness, user satisfaction, mental models, curiosity, trust, and human-ai performance", *Frontiers of Computer Science, Sec. Theoretical Computer Science, 06 February 2023*, vol. 5, 2023, https://doi.org/10.3389/fcomp.2023.1096257.

[34] A. Holzinger, A. M. Carrington, and H. Müller, "Measuring the quality of explanations: The system causability scale (SCS). comparing human and machine explanations", *Computing Research Repository*, vol. abs/1912.09024, 2019.

[35] F. Sovrano and F. Vitali, "An objective metric for explainable AI: how and why to estimate the degree of explainability", *Computing Research Repository*, vol. abs/2109.05327, 2021.

[36] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead", *Computing Research Repository*, vol. abs/1811.10154, 2019.

[37] H. Löfström, K. Hammar, and U. Johansson. *A Meta Survey of Quality Evaluation Criteria in Explanation Methods*, 55–63. Springer International Publishing, 2022.

[38] J. Zhou, A. H. Gandomi, F. Chen, and A. Holzinger, "Evaluating the quality of machine learning explanations: A survey on methods and metrics", *Electronics*, vol. 10, no. 5, 2021, 10.3390/electronics10050593.

[39] M. Tamajka. "How to measure the quality of explanations of AI predictions". https://kinit.sk/how-to-measure-the-quality-of-explanations-of-ai-predictions/,2022.

[40] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A review of machine learning interpretability methods", *Entropy*, vol. 23, no. 1, 2021, 10.3390/e23010018.

[41] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors". In: K. Toutanova and H. Wu, eds., *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Baltimore, Maryland, 2014, 238–247, 10.3115/v1/P14-1023.

[42] M. A. Kłopotek, S. T. Wierzchoń, B. Starosta, D. Czerski, and P. Borkowski, "A method for handling negative similarities in explainable graph spectral clustering of text documents – Extended Version", *Computing Research Repository*, vol. abs/2504.12360, 2025, https://doi.org/10.48550/arXiv.2504.12360.

[43] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead", *Nature Machine Intelligence*, vol. 1, no. 5, 2019, 206–215, 10.1038/S42256-019-0048-X.

[44] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A review of machine learning interpretability methods", *Entropy 2021*, vol. 23, no. 1, 2021, https://doi.org/10.3390/e23010018.

[45] S. Farquhar, J. Kossen, L. Kuhn, et al., "Detecting hallucinations in large language models using semantic entropy", *Nature*, vol. 630, 2024, 625–630, https://doi.org/10.1038/s41586-024-07421-0.

Appendix

A. Results of Clustering Using k-Means

For the completeness of our research, we posed the question of whether it makes sense to use a clustering method different from the straightforward usage of the popular k-means algorithm, that is, whether it makes sense to apply Graph Spectral Clustering. In table 6 we show clustering results of dataset 0 using k-means directly in the respective embedding space. If we compare this with Table 1, which shows clustering results using GSC clustering, we see that GSC offers an improvement in clustering quality. Hence, it makes sense to bother about explainability in GSC in spite of the fact that for the plain k-means, the explanations are straightforward.

Tables 7 - 10 present clustering results for datasets 1 - 4 respectively for k-means clustering. A comparison with tables 2 - 5 allows to draw similar conclusions: GSC offers generally an improvement in clustering quality especially for BERT type embeddings. However, a drawback for some BERT implemnentations is visible. Under some embeddings GSC clustering cannot be computed due to massive negative similarities while k-means clustering is possible in such cases. This issue is subject of our further investigations. As shown in [42], this issue can be successfully mitigated for Glove type embeddings and we are currently experimenting with similar methods for BERT type embeddings, with promising results.

Table 6. k-means clustering of Dataset 0 using various embeddings (vectorizers)

Vectorizer	F1-avg
CountVectorizer	0.249608
TfVectorizer	0.249584
TfidfVectorizer	0.348829
GloVe@wiki	0.285086
GloVe@twitter	0.239426
sBERT@all-MiniLM-L6-v2	0.961594
sBERT@all-MiniLM-L12-v2	0.937874
sBERT@all-mpnet-base-v2	0.925444
sBERT@all-distilroberta-v1	0.929055
sBERT@multi-qa-MiniLM-L6-cos-v1	0.941966
sBERT@multi-qa-distilbert-cos-v1	0.952534
sBERT@multi-qa-mpnet-base-dot-v1	0.762779
BERT@bert-base-uncased#[CLS]	0.346147
BERT@vinai/bertweet-base#[CLS]	0.492142
BERT@distilbert-base-uncased#[CLS]	0.445644
BERT@cardiffnlp/twitter-roberta-base#[CLS]	0.540962
BERT@bert-base-uncased#T_AVG	0.401162
BERT@vinai/bertweet-base#T_AVG	0.369402
BERT@distilbert-base-uncased#T_AVG	0.400333
BERT@cardiffnlp/twitter-roberta-base#T_AVG	0.362900

Table 7. k-means clustering of Dataset 1 using various embeddings (vectorizers)

Vectorizer	F1-avg
CountVectorizer	0.325411
TfVectorizer	0.325468
TfidfVectorizer	0.467889
GloVe@wiki	0.183522
GloVe@twitter	0.212565
sBERT@all-MiniLM-L6-v2	0.890326
sBERT@all-MiniLM-L12-v2	0.912272
sBERT@all-mpnet-base-v2	0.729900
sBERT@all-distilroberta-v1	0.821762
sBERT@multi-qa-MiniLM-L6-cos-v1	0.862951
sBERT@multi-qa-distilbert-cos-v1	0.816841
sBERT@multi-qa-mpnet-base-dot-v1	0.797446
BERT@bert-base-uncased#[CLS]	0.389700
BERT@vinai/bertweet-base#[CLS]	0.558415
BERT@distilbert-base-uncased#[CLS]	0.533903
BERT@cardiffnlp/twitter-roberta-base#[CLS]	0.707646
BERT@bert-base-uncased#T_AVG	0.505968
BERT@vinai/bertweet-base#T_AVG	0.570867
BERT@distilbert-base-uncased#T_AVG	0.529057
BERT@cardiffnlp/twitter-roberta-base#T_AVG	0.531740

Table 8. k-means clustering of Dataset 2 using various embeddings (vectorizers)

Vectorizer	F1-avg
CountVectorizer	0.244935
TfVectorizer	0.245068
TfidfVectorizer	0.470825
GloVe@wiki	0.211871
GloVe@twitter	0.257164
sBERT@all-MiniLM-L6-v2	0.811696
sBERT@all-MiniLM-L12-v2	0.788610
sBERT@all-mpnet-base-v2	0.631467
sBERT@all-distilroberta-v1	0.657888
sBERT@multi-qa-MiniLM-L6-cos-v1	0.715318
sBERT@multi-qa-distilbert-cos-v1	0.586079
sBERT@multi-qa-mpnet-base-dot-v1	0.722792
BERT@bert-base-uncased#[CLS]	0.376012
BERT@vinai/bertweet-base#[CLS]	0.403862
BERT@distilbert-base-uncased#[CLS]	0.363897
BERT@cardiffnlp/twitter-roberta-base#[CLS]	0.586548
BERT@bert-base-uncased#T_AVG	0.393337
BERT@vinai/bertweet-base#T_AVG	0.363000
BERT@distilbert-base-uncased#T_AVG	0.391392
BERT@cardiffnlp/twitter-roberta-base#T_AVG	0.400820

Table 9. k-means clustering of Dataset 3 using various embeddings (vectorizers)

Vectorizer	F1-avg
CountVectorizer	0.239606
TfVectorizer	0.239582
TfidfVectorizer	0.374237
GloVe@wiki	0.187456
GloVe@twitter	0.192742
sBERT@all-MiniLM-L6-v2	0.984532
sBERT@all-MiniLM-L12-v2	0.986152
sBERT@all-mpnet-base-v2	0.990348
sBERT@all-distilroberta-v1	0.974033
sBERT@multi-qa-MiniLM-L6-cos-v1	0.945448
sBERT@multi-qa-distilbert-cos-v1	0.957767
sBERT@multi-qa-mpnet-base-dot-v1	0.972517
BERT@bert-base-uncased#[CLS]	0.329817
BERT@vinai/bertweet-base#[CLS]	0.365470
BERT@distilbert-base-uncased#[CLS]	0.453654
BERT@cardiffnlp/twitter-roberta-base#[CLS]	0.672232
BERT@bert-base-uncased#T_AVG	0.442688
BERT@vinai/bertweet-base#T_AVG	0.508783
BERT@distilbert-base-uncased#T_AVG	0.438658
BERT@cardiffnlp/twitter-roberta-base#T_AVG	0.572831

Table 10. k-means clustering of Dataset 4 using various embeddings (vectorizers)

Vectorizer	F1-avg
CountVectorizer	0.286061
TfVectorizer	0.286051
TfidfVectorizer	0.311880
GloVe@wiki	0.345985
GloVe@twitter	0.239728
sBERT@all-MiniLM-L6-v2	0.715959
sBERT@all-MiniLM-L12-v2	0.708030
sBERT@all-mpnet-base-v2	0.717453
sBERT@all-distilroberta-v1	0.780480
sBERT@multi-qa-MiniLM-L6-cos-v1	0.683618
sBERT@multi-qa-distilbert-cos-v1	0.736226
sBERT@multi-qa-mpnet-base-dot-v1	0.761548
BERT@bert-base-uncased#[CLS]	0.410637
BERT@vinai/bertweet-base#[CLS]	0.399653
BERT@distilbert-base-uncased#[CLS]	0.516114
BERT@cardiffnlp/twitter-roberta-base#[CLS]	0.709762
BERT@bert-base-uncased#T_AVG	0.531288
BERT@vinai/bertweet-base#T_AVG	0.432609
BERT@distilbert-base-uncased#T_AVG	0.523886
BERT@cardiffnlp/twitter-roberta-base#T_AVG	0.505151