Journal of Classification
https://doi.org/10.1007/s00357-026-09544-7

ORIGINAL RESEARCH

Spherical Double K-Means: A Co-clustering Approach for Textual Data Analysis

Ilaria Bombelli[1] · **Domenica Fioredistella Iezzi**[2] · **Emiliano Seri**[2] · **Maurizio Vichi**[3]

Received: 24 February 2025 / Accepted: 19 February 2026

Abstract
In text analysis, spherical k-means (SKM) is a specialized k-means clustering algorithm widely utilized for grouping documents represented in high-dimensional, sparse term-document matrices, often normalized using techniques like TF-IDF. Researchers frequently seek to cluster not only documents but also the terms associated with them into coherent groups. To address this dual clustering requirement, we introduce spherical double k-means (SDKM), a novel methodology that simultaneously clusters documents and terms. This methodology offers several advantages, such as enabling more effective topic identification and keyword extraction, enhancing interpretability, computational efficiency, and efficiency in capturing dynamic changes in thematic content over time. It also facilitates the uncovering of nuanced patterns and structures of textual data. We apply SDKM to simulated and real data. The real data applications are on the corpus of US presidential inaugural addresses, spanning from George Washington in 1789 to Joe Biden in 2021, and to the 20 Newsgroups corpus. Our analysis reveals distinct clusters of words and documents that correspond to significant themes and periods, showcasing the method's ability to facilitate a deeper understanding of the data. Our findings demonstrate the efficacy of SDKM in uncovering underlying patterns in textual data.

Keywords Textual data · Co-clustering · Topic modeling · Spherical double k-means

✉ Emiliano Seri
emiliano.seri@uniroma2.it

Ilaria Bombelli
ilaria.bombelli@istat.it

Domenica Fioredistella Iezzi
stella.iezzi@uniroma2.it

Maurizio Vichi
maurizio.vichi@uniroma1.it

[1] Directorate for Methodology and Statistical Process Design, Italian National Institute of Statistics, Via Cesare Balbo 16, 00184 Rome, Italy

[2] Department of Enterprise Engineering "Mario Lucertini", University of Rome Tor Vergata, Via del Politecnico 1, 00133 Rome, Lazio, Italy

[3] Department of Statistics, Sapienza University of Rome, Piazzale Aldo Moro 5, 00185 Rome, Lazio, Italy

Published online: 06 March 2026

1 Introduction

In recent years, the rapid increase in textual data from various sources, such as social networks, electronic health records, insurance claims, and news outlets, has made effective text analysis methods more critical than ever. Text clustering, the process of grouping similar texts, is crucial in this analysis. Text clustering, which involves grouping similar documents without predefined categories or labels (Iezzi, 2012), plays a pivotal role in uncovering hidden patterns, topics, or clusters within large corpora. This facilitates the exploration and organization of vast textual datasets based on inherent similarities, aiding in information retrieval, topic modeling, and data summarization. Text clustering spans a broad set of methods, including partition-based approaches such as k-means and spherical k-means, hierarchical clustering, spectral/co-clustering on bipartite graphs, matrix-factorization methods (e.g., NMF), and probabilistic topic models (pLSA/LDA). Most of the existing methods are reviewed in Mehta et al. (2024).

Among the most used clustering algorithms, k-means and its variants are valued for their simplicity, scalability, and ability to classify large and unstructured data quickly. However, traditional k-means clustering has limitations when applied to textual data. The algorithm's reliance on Euclidean distance often fails to accurately represent relationships in high-dimensional spaces, such as those created by text representations like term frequency–inverse document frequency (TF-IDF) or bag-of-words, ignoring meaningful semantic relationships like synonyms and context. Additionally, the high dimensionality of textual data can render Euclidean distance less meaningful, leading to ineffective clustering results. Another challenge is k-means' sensitivity to initialization, as its performance heavily depends on the number of clusters and the initial placement of centroids, which can result in unstable or suboptimal outcomes. The algorithm also assumes that clusters are spherical and evenly distributed, a condition rarely met in textual data, which often forms non-convex or overlapping clusters. Furthermore, k-means treats text as numerical data, lacking semantic understanding, which can lead to similar words being assigned to different clusters. Lastly, k-means is sensitive to outliers and noise, such as rare words or misspellings, distorting cluster centroids and reducing overall clustering quality. Moreover, textual data often exhibit high dimensionality, sparsity, and noise, making the Euclidean distance metric used in standard k-means less suitable. This is because Euclidean distance can overemphasize longer documents with higher term weights, leading to biased clustering results (Hornik et al., 2012).

To address these challenges, alternative distance measures like cosine similarity have been employed in text clustering. Cosine similarity assesses the similarity between two vectors based on the cosine of the angle between them Maher and Joshi (2016).

To better clarify this aspect, cosine similarity is widely adopted in text mining due to the high-dimensional and sparse nature of textual data (Huang et al., 2008). In a term-document matrix, most entries are zero, and document lengths can vary significantly. Unlike Euclidean distance, which is sensitive to vector magnitudes and can be distorted by document length or varying word counts, cosine similarity measures the orientation between vectors, effectively normalizing for length. This makes it more robust in comparing documents based on content rather than size. In addition, Euclidean distance for textual data often fails to accurately represent relationships in high-dimensional spaces, such as those created by text representations like TF-IDF or bag-of-words, and ignores meaningful semantic relationships

like synonyms and context. Moreover, in sparse, high-dimensional spaces, Euclidean distance tends to suffer from the "curse of dimensionality," where distances become less meaningful and discriminative (Aggarwal et al., 2001). Cosine similarity mitigates this by focusing on the angle between vectors, providing a more reliable similarity measure for clustering and classification tasks in natural language processing.

In this paper, we introduce the spherical double k-means (SDKM) clustering method, an original approach for the simultaneous partitioning of terms and documents. SDKM integrates the principles of double k-means clustering (DKM; Vichi, 2001) and spherical k-means (SKM; Dhillon and Modha, 2001; Hornik et al., 2012), tailored specifically for textual data analysis. By incorporating cosine similarity into the co-clustering framework, SDKM effectively handles high dimensionality, sparsity, and noise, facilitating the discovery of meaningful patterns in text corpora.

Co-clustering, or the simultaneous clustering of rows and columns in a term-document matrix, allows for the classification of terms based on the entire set of documents and vice versa (Celardo et al., 2016). Simultaneous consideration of row and column clusters enhances robustness by reducing the impact of sparsity and noise, as the co-clustering process leverages shared structure across both dimensions. This results in more stable and interpretable clusters, particularly when dealing with large, sparse, and non-negative matrices typical in text analysis. The interrelated nature of row and column clusters in co-clustering methods fosters a more comprehensive understanding of the underlying data structure.

Co-clustering is particularly needed in this context because it allows for the simultaneous grouping of both terms and documents, capturing the interdependencies between them. This joint clustering approach leads to more robust and interpretable results compared to methods that cluster terms or documents separately. Theoretically, co-clustering exploits the shared structure of the term-document matrix, mitigating the effects of noise and sparsity inherent in textual data. Practically, this improves the quality of downstream tasks such as topic modeling, sentiment analysis, and information retrieval by revealing latent patterns that might otherwise remain hidden. Our proposed spherical double k-means (SDKM) method innovates by integrating cosine similarity within the co-clustering framework, tailored specifically for high-dimensional and sparse text data, thus offering a novel and effective approach for textual data analysis.

The proposed SDKM methodology offers a versatile approach with broad applicability in various tasks, including topic modeling, document clustering, sentiment analysis, information retrieval, and document summarization. Its ability to capture the inherent structure of textual data makes it particularly well-suited for extracting meaningful insights from large text corpora.

We demonstrate the effectiveness of SDKM through its application to the US presidential inaugural addresses, a corpus spanning from George Washington in 1789 to Joe Biden in 2021, freely available in the R package `quanteda` (Benoit et al., 2018). By analyzing this dataset, we uncover thematic evolutions and patterns in presidential rhetoric over more than two centuries.

The remainder of the paper is organized as follows. Section 2 provides background on textual data processing and reviews relevant clustering methodologies. Section 3 details the proposed SDKM methodology, and Sect. 4 reports computational choices and empirical scaling. Section 5 presents the data generation mechanism, the pseudo-F index for selecting

(K, Q), and a simulation study assessing partition and centroid recovery. Section 6 applies SDKM to the U.S. presidential inaugural addresses. In Sect. 7, we compare SDKM with DKM on the same corpus. In Sect. 8, we benchmark SDKM against mixtures of von Mises–Fisher components on both the real corpus and simulated scenarios. Sect. 9 reports a second application to the 20 Newsgroups corpus. Finally, Sect. 10 concludes and outlines avenues for future research. Appendix 1 collect the monotonicity proof, while the detailed timing tables and the full pseudo-F grids are included in the supplementary material.

2 Background

2.1 Addressing Textual Data

In recent years, the exponential growth of text-based information, ranging from social networks and electronic health records to insurance claims and news outlets, has highlighted the necessity of robust text analysis tools. Unlike structured numerical data, textual data is inherently unstructured, high-dimensional, and prone to significant noise and sparsity (Churchill & Singh, 2022; Iezzi & Celardo, 2020). This complexity demands specialized methods of data representation, preprocessing, and clustering to effectively manage large text corpora (Celardo et al., 2016). A foundational step in text analysis is transforming unstructured documents into a numeric format. A widely adopted solution is the vector space model (VSM), which represents each document as a vector of term frequencies. Although the VSM is conceptually straightforward and often effective, it introduces complications related to document length and high dimensionality (Aggarwal & Zhai, 2012). One way to mitigate these effects is by normalizing term vectors or employing similarity measures such as cosine similarity, which can reduce the bias toward longer documents (Hornik et al., 2012).

Before converting documents to numeric vectors, the data undergoes a preprocessing phase (Cozzolino & Ferraro, 2022) that typically includes tokenization (splitting text into unigrams or bigrams), stopword removal (removing common, low-information words, e.g., "the," "and," "or," to reduce dimensionality), pruning (discarding terms that appear too frequently or too rarely, thereby reducing noise), and stemming or lemmatization (stemming reduces words to their root form using heuristic rules, whereas lemmatization maps words to their dictionary base form; for example, "studies," "studying," and "studied" become "study"). By trimming the vocabulary size and normalizing word variants, these steps reduce dimensionality and noise, thereby improving clustering quality and computational efficiency.

Nonetheless, textual data remains difficult to analyze due to its high sparsity: most cells in a term-document matrix are zero, and the vocabulary can easily scale to tens or hundreds of thousands of terms. As a result, standard clustering methods, such as k-means (Mcqueen, 1967), may become ineffective and may yield to poor performance when applied to high-dimensional and sparse text data. These methods often rely on distance metrics like Euclidean distance, which can lose discriminative power in such settings and fail to capture the complex structure of textual information. Further complicating matters, the same concept may appear in numerous synonyms or domain-specific slang, making purely frequency-based techniques inadequate for capturing deeper semantic relationships (Lodhi et al., 2002). Document length is also highly variable; some texts may contain only a few words, whereas others include thousands. Traditional clustering approaches relying on Euclidean metrics thus risk

overweighting longer documents (Dhillon & Modha, 2001), so cosine-based measures often prove more appropriate for text-based scenarios (Hornik et al., 2012; Zhao & Karypis, 2004).

Many mainstream clustering algorithms have been adapted or extended to reduce the impact of high dimensionality and sparsity in text. Prototype-based methods (e.g., spherical k-means) modify k-means to incorporate cosine similarity, improving results on high-dimensional text corpora (Dhillon & Modha, 2001); however, they typically cluster documents only and do not jointly cluster terms and documents as SDKM does. Graph-based methods (e.g., spectral clustering) represent documents as nodes in a graph, with edges weighted by kernel- or substring-based similarities (Di Nuzzo & Ingrassia, 2022; Janani & Vijayarani, 2019; Lodhi et al., 2002), but they often lack the explicit dual clustering of terms and documents. Hierarchical approaches (e.g., agglomerative and divisive methods) build nested clusters (Steinbach et al., 2000), which can be beneficial for smaller corpora or when interpretable dendrograms are desired; nonetheless, they can struggle with scalability and usually do not incorporate simultaneous clustering of terms and documents. Finally, model-based approaches (e.g., latent Dirichlet allocation) assume documents arise from mixtures of underlying distributions: while Gaussian mixture models must address text sparsity and manifold structure (Bouveyron et al., 2019), latent Dirichlet allocation (LDA) specializes in revealing latent topics by modeling each document as a mixture of hidden themes (Blei et al., 2003); however, such approaches do not explicitly consider the co-clustering of terms and documents using geometric similarity measures.

Our proposed spherical double k-means (SDKM) method differs fundamentally from these approaches in terms of data assumptions, clustering granularity, and similarity metrics. SDKM explicitly performs co-clustering of both terms and documents by integrating cosine similarity within a double k-means framework. This simultaneous partitioning captures the interdependencies between rows and columns in the term-document matrix, enabling a more flexible and robust clustering that effectively handles sparsity and high dimensionality without relying on probabilistic models or graph structures. This approach provides a novel and practical solution for extracting meaningful patterns from large textual corpora.

2.2 Spherical K-Means and Double K-Means Methodologies

Our methodological proposal integrates two well-known clustering techniques: the former, DKM, is an extension of k-means (KM) and it is used to simultaneously cluster units and variables of a data matrix; the latter, SKM, is a useful tool to cluster units of a data matrix, as the well-known KM algorithm does; however, the main difference between SKM and KM lies in the fact that SKM uses a non-euclidean distance to compute dissimilarities between any pair of units. In the following, a formal definition and detailed description on the two techniques are provided.

2.2.1 Double K-Means: DKM

Given a data matrix $\boldsymbol{X} = \{x_{ij} : i = 1, \ldots, N, j = 1, \ldots, J\}$ where i indicates units and j indicates variables, the DKM algorithm (Vichi, 2001) simultaneously partitions units into K clusters and variables into Q clusters. In the context of textual data, the units (i.e., the rows of $\boldsymbol{X}$) correspond to terms, while the variables (i.e., the columns) correspond to documents. Therefore, the final outputs of the algorithm are two membership matrix modeling

the partitions of units and variables into clusters, and a centroids matrix of dimension $K \times Q$, synthesizing the subsets of $\boldsymbol{X}$. More in details, the whole data matrix is partitioned into subsets (blocks) and each block is represented by one centroid. The DKM model is formally defined as follows:

$$\boldsymbol{X} = \boldsymbol{U}\overline{\boldsymbol{Y}}\boldsymbol{V}' + \boldsymbol{E}, \tag{1}$$

subject to the constraints:

$$\begin{aligned} & u_{ik} \in \{0, 1\}, \quad \forall i = 1, \ldots, N; k = 1, \ldots, K; \\ & \sum_{k=1}^{K} u_{ik} = 1, \quad \forall i = 1, \ldots, N; \\ & v_{jq} \in \{0, 1\}, \quad \forall j = 1, \ldots, J; q = 1, \ldots, Q; \\ & \sum_{q=1}^{Q} v_{jq} = 1, \quad \forall j = 1, \ldots, J, \end{aligned}$$

where $\boldsymbol{U}$ is the $N \times K$ membership matrix modelling the partition of the N units inside K clusters; $\boldsymbol{V}$ is the $J \times Q$ membership matrix modelling the partition of the J variables inside Q clusters; $\overline{\boldsymbol{Y}}$ is the $K \times Q$ centroids matrix, whose element $\overline{y}_{kq}$ synthesizes the observations within the block identified by the units belonging to the k-th cluster and the variables belonging to the q-th cluster.

The centroid matrix $\overline{\boldsymbol{Y}}$ thus synthesizes the most relevant information of the data matrix $\boldsymbol{X}$ and can be considered as a reduced data matrix, with rank at most equal to the minimum between K and Q. The matrix $\boldsymbol{E}$ is the $N \times J$ matrix of errors. Finally, the constraints $u_{ik} \in \{0, 1\}$ and $\sum_{k=1}^{K} u_{ik} = 1$ ensure that $\boldsymbol{U}$ is a binary and row-stochastic matrix; similarly, the constraints $v_{jq} \in \{0, 1\}$ and $\sum_{q=1}^{Q} v_{jq} = 1$ ensure that $\boldsymbol{V}$ is a binary and row-stochastic matrix.

Remark 1 The centroids matrix is updated as follows:

$$\overline{\boldsymbol{Y}} = \boldsymbol{U}^{+}\boldsymbol{X}\boldsymbol{V}^{+\prime} = (\boldsymbol{U}'\boldsymbol{U})^{-1}\boldsymbol{U}\boldsymbol{X}\boldsymbol{V}(\boldsymbol{V}'\boldsymbol{V})^{-1}, \tag{2}$$

where $\boldsymbol{U}^{+} = (\boldsymbol{U}'\boldsymbol{U})^{-1}\boldsymbol{U}$ and $\boldsymbol{V}^{+} = (\boldsymbol{V}'\boldsymbol{V})^{-1}\boldsymbol{V}'$ denote the Moore-Penrose inverse of the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively.

Remark 2 If the units' membership matrix $\boldsymbol{U}$ degenerates into an identity matrix of order N, i.e., $\boldsymbol{U} = \boldsymbol{1}_K$, the model 1 can be written as $\boldsymbol{X} = \overline{\boldsymbol{Y}}\overline{\boldsymbol{V}}' + \boldsymbol{E}$, which is the model of the KM to partition variables into Q clusters.
Similarly, if the variables' membership matrix $\boldsymbol{V}$ degenerates into an identity matrix of order J, i.e., $\boldsymbol{V} = \boldsymbol{1}_J$, the model 1 can be written as $\boldsymbol{X} = \boldsymbol{U}\overline{\boldsymbol{Y}} + \boldsymbol{E}$, which is the model of the KM to partition units into K clusters.

2.2.2 Spherical K-Means: SKM

Let $\boldsymbol{X}$ be the term-document matrix as defined in Sect. 2.2.1. Given $\boldsymbol{X}$, SKM partitions units into K clusters. First proposed by Dhillon and Modha (2001) and then specified in Hornik et al. (2012), what makes it different from the KM algorithm, whose objective is the same, is the use of the distance function to compute dissimilarities between any pair of units: indeed, the standard KM uses Euclidean distance, while the SKM computes cosine dissimilarity.

Definition 1 Given two vectors $\boldsymbol{a}$ and $\boldsymbol{b} \in \mathbb{R}^n$ the cosine dissimilarity between them is defined as follows:

$$\mathrm{d}(\boldsymbol{a}, \boldsymbol{b}) = 1 - \cos(\boldsymbol{a}, \boldsymbol{b}) = 1 - \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{\|\boldsymbol{a}\| \|\boldsymbol{b}\|}. \tag{3}$$

The cosine dissimilarity takes into account the angle between the two vectors.
If the two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are normalized, i.e., $\|a\| = \|b\| = 1$, then the cosine dissimilarity becomes:

$$\mathrm{d}(\boldsymbol{a}, \boldsymbol{b}) = 1 - \cos(\boldsymbol{a}, \boldsymbol{b}) = 1 - \langle \boldsymbol{a}, \boldsymbol{b} \rangle. \tag{4}$$

Cosine dissimilarity takes the value 0 if the vectors are exactly equal, while it takes the value 1 if the two vectors do not share any elements.

The SKM algorithm aims to minimize the (cosine) dissimilarity between observations and centroids.[1] Therefore, the objective function of the SKM is formalized as follows:

$$f(\boldsymbol{U}, \overline{\boldsymbol{X}}) = \sum_{i=1}^{N} \sum_{k=1}^{K} u_{ik} (1 - \cos(\boldsymbol{x}_i, \overline{\boldsymbol{x}_k})) = \sum_{i=1}^{N} \sum_{k=1}^{K} u_{ik} \Big(1 - \frac{\langle \boldsymbol{x_i}, \overline{\boldsymbol{x_k}} \rangle}{\|\boldsymbol{x_i}\| |\overline{\boldsymbol{x_k}}\|} \Big), \tag{5}$$

s.t.

$$u_{ik} \in \{0, 1\}, \quad \forall\, i = 1, \ldots, N; k = 1, \ldots, K;$$
$$\sum_{k=1}^{K} u_{ik} = 1, \quad \forall\, i = 1, \ldots, N,$$

where $u_{ik} \in \{0, 1\}$ is the membership of unit i to cluster k and $\overline{\boldsymbol{x}_k}$ is the k-th centroid, synthesizing the observations (units) belonging to the k-th cluster.

Remark 3 Likewise, instead of minimizing Eq. 5, we can maximize

$$f(\boldsymbol{U}, \overline{\boldsymbol{X}}) = \sum_{i=1}^{N} \sum_{k=1}^{K} u_{ik}\, s_{ik}, \tag{6}$$

where $s_{ik} = \cos(\boldsymbol{x}_i, \overline{\boldsymbol{x}}_k) = \frac{\langle \boldsymbol{x}_i, \overline{\boldsymbol{x}}_k \rangle}{\|\boldsymbol{x}_i\| \, \|\overline{\boldsymbol{x}}_k\|}$.

The objective function in Eq. 6 can be also written in a matrix form notation.

Let $\boldsymbol{M}$ be a $N \times N$ diagonal matrix having on the main diagonal the inverse of the norm of the rows of the data matrix $\boldsymbol{X}$, defined as follows:

$$\boldsymbol{M} = \begin{pmatrix} \frac{1}{\|\boldsymbol{x}_1\|} & 0 & \ldots & 0 \\ 0 & \frac{1}{\|\boldsymbol{x}_2\|} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & \ldots & \frac{1}{\|\boldsymbol{x}_N\|} \end{pmatrix} = (\mathrm{diag}(\boldsymbol{X}\boldsymbol{X}'))^{-\frac{1}{2}}.$$

[1] Although cosine dissimilarity is not a true distance metric in the mathematical sense, since it does not satisfy the triangle inequality, it remains a valid and effective dissimilarity measure in many applications involving high-dimensional data, especially in text mining (Huang et al., 2008).

Let $\boldsymbol{W}$ be a $K \times K$ diagonal matrix having on the main diagonal the inverse of the norm of the rows of the centroid data $\overline{\boldsymbol{X}}$:

$$\boldsymbol{W} = \begin{pmatrix} \frac{1}{\|\overline{\boldsymbol{x}}_1\|} & 0 & \dots & 0 \\ 0 & \frac{1}{\|\overline{\boldsymbol{x}}_2\|} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \frac{1}{\|\overline{\boldsymbol{x}}_K\|} \end{pmatrix} = \left(\text{diag}(\overline{\boldsymbol{X}}\,\overline{\boldsymbol{X}}') \right)^{-\frac{1}{2}}.$$

Then, the objective function Eq. 6 to be maximized can be reformulated in matrix form as follows:

$$f(\boldsymbol{U}, \overline{\boldsymbol{X}}) = \text{tr}(\boldsymbol{M}\boldsymbol{X}\overline{\boldsymbol{X}}'\boldsymbol{W}\boldsymbol{U}'),$$

where $\text{tr}(\cdot)$ is the trace operator. It has to be noted that $\boldsymbol{S} = \boldsymbol{X}\overline{\boldsymbol{X}}'\boldsymbol{U}'$ is the $N \times N$ matrix modelling similarities between the data matrix $\boldsymbol{X}$ and the centroids matrix $\overline{\boldsymbol{X}}$ and having on the main diagonal the similarities between the N observations and their related centroids. More in detail, given a unit i belonging to cluster k, then s_{ik} denotes the similarity between observation $\boldsymbol{x}_i$ and the centroid $\overline{\boldsymbol{x}}_k$.

If the data matrix $\boldsymbol{X}$ and the centroids matrix $\overline{\boldsymbol{X}}$ are normalized by row, then the objective function becomes:

$$f(\boldsymbol{U}, \overline{\boldsymbol{X}}) = \text{tr}(\boldsymbol{X}\overline{\boldsymbol{X}}'\boldsymbol{U}') = \text{tr}(\boldsymbol{X}'\boldsymbol{U}\overline{\boldsymbol{X}}),$$

where the last equality is given by the trace operator property $\text{tr}(\boldsymbol{A}\boldsymbol{B}') = \text{tr}(\boldsymbol{A}'\boldsymbol{B})$. The normalized version of the objective function, i.e., an objective function which is upper bounded by 1, is the following:

$$f(\boldsymbol{U}, \overline{\boldsymbol{X}}) = \frac{\text{tr}(\boldsymbol{X}'\boldsymbol{U}\overline{\boldsymbol{X}})}{\sqrt{\text{tr}(\boldsymbol{X}'\boldsymbol{X}) \cdot \text{tr}(\overline{\boldsymbol{X}}'\boldsymbol{U}'\boldsymbol{U}\overline{\boldsymbol{X}})}}.$$

For the sake of notation, we let $\boldsymbol{X}_t = \boldsymbol{U}\overline{\boldsymbol{X}}$ be the theoretical matrix of the SKM model. Then, the objective function can be written in a more compact way as follows:

$$f(\boldsymbol{X}_t) = \frac{\text{tr}(\boldsymbol{X}'\boldsymbol{X}_t)}{\sqrt{\text{tr}(\boldsymbol{X}'\boldsymbol{X}) \cdot \text{tr}(\boldsymbol{X}_t'\boldsymbol{X}_t)}}. \tag{7}$$

It has to be noted that the objective function in Eq. 7 is upper bounded by 1, and it is equal to 1 when the data matrix $\boldsymbol{X}$ matches the theoretical one $\boldsymbol{X}_t$.

In the following, the updating formula of the centroids matrix $\overline{\boldsymbol{X}}$ and partition $\boldsymbol{U}$ will be derived (Eqs. 9 and 10) and we will assume that the data matrix $\boldsymbol{X}$ and the centroids matrix $\overline{\boldsymbol{X}}$ are normalized by row, and we denote them as $\boldsymbol{X}^{\text{N}}$ and $\overline{\boldsymbol{X}}^{\text{N}}$.

In order to maximize the objective function Eq. 7, it can be useful to consider the problem row-wise: for every row, it is needed to maximize the similarity between observation $\boldsymbol{X}_i^{\text{N}}$ and observation $\overline{\boldsymbol{x}}_k^N$, where k denotes the cluster to which unit i belongs. As it is usually done when maximizing a function, the maximum can be found by setting equal to zero the first derivative, find the root of the equation, and then verify that the root is indeed a maximum. Let θ denote the angle between $\boldsymbol{X}_i^{\text{N}}$ and $\overline{\boldsymbol{x}}_k^N$, let $f(\theta) = \cos(\theta)$ be the function to be maximized. Then, $\frac{df}{d\theta} = -\sin(\theta) = 0 \iff \sin(\theta) = 0$. Therefore, we can conclude that the objective function is maximized when the angle θ between $\boldsymbol{X}_i^{\text{N}}$ and $\overline{\boldsymbol{x}}_k^N$ is a 0 or 180 degrees angle, namely when the two vectors $\boldsymbol{X}_i^{\text{N}}$ and $\overline{\boldsymbol{x}}_k^N$ coincide or are parallel to each other. Formally, the function is maximized when $\boldsymbol{X}_i^{\text{N}} = c \cdot \overline{\boldsymbol{x}}_k^N$, $c \in \mathbb{R}$.

By extending the reasoning to the whole data matrix $\boldsymbol{X}^{\mathrm{N}}$, the objective function is maximized when each of the row-vectors of the data matrix $\boldsymbol{X}^{\mathrm{N}}$ is proportional to the centroids row-vector $\overline{\boldsymbol{x}}_k^N$, where k is the cluster unit i belongs to. Formally, the objective function is maximized when

$$\boldsymbol{X}^{\mathrm{N}} = (\boldsymbol{C}\boldsymbol{U})\overline{\boldsymbol{X}}^N. \tag{8}$$

Here $\boldsymbol{C}$ is an $N \times N$ diagonal matrix whose diagonal elements are the constants of proportionality between each pair of vectors.

By solving Eq. 8 w.r.t. $\overline{\boldsymbol{X}}^{\mathrm{N}}$, we obtain

$$\boldsymbol{U}'\boldsymbol{X}^{\mathrm{N}} = \boldsymbol{U}'\boldsymbol{C}\boldsymbol{U}\overline{\boldsymbol{X}}^{\mathrm{N}} \iff \overline{\boldsymbol{X}}^{\mathrm{N}} = (\boldsymbol{U}'\boldsymbol{C}\boldsymbol{U})^{-1}\boldsymbol{U}'\boldsymbol{X}^{\mathrm{N}},$$

where firstly we pre-multiply both sides of the equation by $\boldsymbol{U}'$ to make $\boldsymbol{U}'\boldsymbol{C}\boldsymbol{U}$ invertible and then we isolate $\overline{\boldsymbol{X}}^{\mathrm{N}}$.

In addition, since the centroids matrix $\overline{\boldsymbol{X}}^{\mathrm{N}}$ must be normalized by row, we obtain the following updating formula for $\overline{\boldsymbol{X}}^{\mathrm{N}}$:

$$\overline{\boldsymbol{X}}^{\mathrm{N}} = \frac{(\boldsymbol{U}'\boldsymbol{U})^{-1}\boldsymbol{U}'\boldsymbol{X}^{\mathrm{N}}}{\|(\boldsymbol{U}'\boldsymbol{U})^{-1}\boldsymbol{U}'\boldsymbol{X}^{\mathrm{N}}\|}. \tag{9}$$

Finally, the partition of units inside clusters is provided by including the generic unit i into the cluster whose centroid is the closest one according to the cosine similarity. Formally, the partition of units inside cluster k is

$$\pi_k = \{i \text{ s.t. } \boldsymbol{x}_i^{N\prime}\overline{\boldsymbol{X}}_k^{\mathrm{N}} \leq \boldsymbol{x}_i^{N\prime}\overline{\boldsymbol{X}}_l^{\mathrm{N}},\ \forall\, l \in \{1, \ldots, K\}, l \neq k\}. \tag{10}$$

2.2.3 Spherical K-Means for Clustering Textual Data

The main difference between traditional k-means and spherical k-means lies in the choice of distance metric. When evaluating the distance between two vectors using Euclidean distance, high values may result even when the vectors point in the same direction but differ in magnitude. This is problematic in textual applications, as document lengths vary and absolute term frequencies may differ substantially. In contrast, cosine similarity, employed in SKM, measures the angle between vectors rather than their length, focusing on the relative usage proportions of words rather than their absolute counts. This distinction is crucial in text clustering because words and documents are typically highly correlated. Traditional k-means tends to form one or two large dominant clusters along with several residual clusters. This behavior occurs because Euclidean distance is most effective when the features are uncorrelated, a condition rarely met in textual data. Often, researchers pre-process text via factor-analytic techniques to reduce correlation before applying k-means. By directly using cosine similarity, SKM naturally overcomes this limitation, yielding clusters that more accurately reflect underlying thematic proportions. As a result, SKM tends to isolate more semantically coherent topics, capturing nuanced differences in rhetoric that standard Euclidean-based clustering may obscure.

3 Methodology: Spherical Double K-Means

We integrate SKM in DKM to simultaneously cluster units and variables with spherical shape clusters. We therefore introduce the spherical double k-means. SDKM methodology aims to

incorporate the advantages of SKM, i.e., its ability to address over-dispersion in the data and detect noise, into a double clustering scenario.

Let $\boldsymbol{X}$ be the term-document matrix as defined in Sect. 2.2.1. Suppose we want to partition $\boldsymbol{X}$ using K clusters for row profiles and Q clusters for column profiles. Let $\boldsymbol{U}_{N\times K}$ be the matrix of memberships for the row profiles, $\boldsymbol{V}_{J\times Q}$ the matrix of memberships for the column profiles, and $\overline{\boldsymbol{Y}}_{K\times Q}$ the matrix of centroids. Also, let $\boldsymbol{x}_i$ denote row i of $\boldsymbol{X}$, $\boldsymbol{x}_{.j}$ column j of $\boldsymbol{X}$, $\overline{\boldsymbol{y}}_{k.}$ row k of $\overline{\boldsymbol{Y}}$, and $\overline{\boldsymbol{y}}_{.q}$ column q of $\overline{\boldsymbol{Y}}$. Assuming the goal is to find an approximation matrix $\boldsymbol{X}_t$ of $\boldsymbol{X}$, such that:

$$\boldsymbol{X} = \boldsymbol{X}_t + \boldsymbol{E},$$

where $\boldsymbol{X}_t$ is derived from the decomposition of matrix $\boldsymbol{X}$ and is defined as:

$$\boldsymbol{X}_t = \boldsymbol{U}\overline{\boldsymbol{Y}}\boldsymbol{V}'.$$

The cosine similarity between matrices $\boldsymbol{X}$ and $\boldsymbol{X}_t$ can be formulated as:

$$\text{tr}(\boldsymbol{X}'\boldsymbol{X}_t) = \text{tr}(\boldsymbol{X}'\boldsymbol{U}\overline{\boldsymbol{Y}}\boldsymbol{V}') = \text{tr}(\boldsymbol{X}\boldsymbol{V}\overline{\boldsymbol{Y}}\boldsymbol{U}') = \text{tr}(\boldsymbol{X}\boldsymbol{X}_t').$$

The objective is then to maximize this similarity function by solving the following maximization problem:

$$\max_{\boldsymbol{U},\boldsymbol{V},\overline{\boldsymbol{Y}}} f(\boldsymbol{U},\boldsymbol{V},\overline{\boldsymbol{Y}}) = \text{tr}(\boldsymbol{X}'\boldsymbol{U}\overline{\boldsymbol{Y}}\boldsymbol{V}').$$

As in the case of SKM, the centroid matrix that maximizes the objective function is derived by maximizing the cosine of the angle formed by each row of matrix $\boldsymbol{X}$ and the approximated data matrix $\boldsymbol{X}_t$, which is maximized when the two vectors are equal or proportional to each other, i.e., when:

$$\boldsymbol{X} = \boldsymbol{C}\boldsymbol{U}\overline{\boldsymbol{Y}}\boldsymbol{V}',$$

where $\boldsymbol{C}$ is a diagonal matrix containing the proportionality constants. Solving this yields the expression for the centroid matrix $\overline{\boldsymbol{Y}}$:

$$\overline{\boldsymbol{Y}} = (\boldsymbol{U}'\boldsymbol{C}\boldsymbol{U})^{-1}\boldsymbol{U}'\boldsymbol{X}\boldsymbol{V}(\boldsymbol{V}'\boldsymbol{V})^{-1}.$$

Since matrix $\overline{\boldsymbol{Y}}$ must be normalized by rows, it is given by:

$$\overline{\boldsymbol{Y}} = \frac{(\boldsymbol{U}'\boldsymbol{U})^{-1}\boldsymbol{U}'\boldsymbol{X}\boldsymbol{V}(\boldsymbol{V}'\boldsymbol{V})^{-1}}{\|(\boldsymbol{U}'\boldsymbol{U})^{-1}\boldsymbol{U}'\boldsymbol{X}\boldsymbol{V}(\boldsymbol{V}'\boldsymbol{V})^{-1}\|}. \tag{11}$$

The objective function of the spherical double k-means can be normalized:

$$\max_{\boldsymbol{U},\boldsymbol{V},\overline{\boldsymbol{Y}}} f(\boldsymbol{U},\boldsymbol{V},\overline{\boldsymbol{Y}}) = \frac{\text{tr}\left(\boldsymbol{X}'\,\boldsymbol{U}\,\overline{\boldsymbol{Y}}\,\boldsymbol{V}'\right)}{\sqrt{\text{tr}\left(\boldsymbol{X}'\boldsymbol{X}\right)\ \text{tr}\left(\boldsymbol{U}\,\overline{\boldsymbol{Y}}\,\boldsymbol{V}'\,\boldsymbol{V}\,\overline{\boldsymbol{Y}}'\,\boldsymbol{U}'\right)}}. \tag{12}$$

In this way, the objective function $\max_{\boldsymbol{U},\boldsymbol{V},\overline{\boldsymbol{Y}}} f(\boldsymbol{U},\boldsymbol{V},\overline{\boldsymbol{Y}})$ is less than or equal to 1, and it assumes the value 1 when the data matrix $\boldsymbol{X}$ coincides with the approximated matrix $\boldsymbol{X}_t$. It can be observed that if the membership matrix of the variables $\mathbf{V}$ degenerates into an identity matrix of order J, the SDKM problem reduces to the SKM problem concerning the rows. In this case, the objective function becomes:

$$\max_{\boldsymbol{X},\boldsymbol{Y}} \text{tr}(\boldsymbol{X}'\overline{\boldsymbol{Y}}\boldsymbol{V}'),$$

and the centroid matrix becomes ($\overline{\boldsymbol{Y}} = \boldsymbol{U}'\boldsymbol{X}$). Similarly, if the membership matrix of the units $\boldsymbol{U}$ degenerates into an identity matrix of order N, it is necessary to cluster only the

variables of the data matrix, and the maximization problem of the SDKM is reduced to the following:

$$\max_{\boldsymbol{X},\boldsymbol{Y}} \operatorname{tr}(\boldsymbol{X}'\overline{\boldsymbol{Y}}\boldsymbol{V}'),$$

where the centroid matrix is $\overline{\boldsymbol{Y}} = \boldsymbol{X}\boldsymbol{V}$.

For fixed $(\boldsymbol{V}, \overline{\boldsymbol{Y}})$, we define row-cluster centroids in term space

$$\boldsymbol{c}_k \;=\; \overline{\boldsymbol{y}}_{k\cdot}\,\boldsymbol{V}' \in \mathbb{R}^J, \qquad k = 1, \ldots, K. \tag{13}$$

For fixed $(\boldsymbol{U}, \overline{\boldsymbol{Y}})$, we define column-cluster centroids in document space

$$\boldsymbol{d}_q \;=\; \boldsymbol{U}\,\overline{\boldsymbol{y}}_{\cdot q} \in \mathbb{R}^N, \qquad q = 1, \ldots, Q. \tag{14}$$

With $\cos(\boldsymbol{a}, \boldsymbol{b}) = \langle \boldsymbol{a}, \boldsymbol{b}\rangle / (\|\boldsymbol{a}\|\,\|\boldsymbol{b}\|)$, the updates are:

$$u_{ik} \;=\; \begin{cases} 1, & \text{if } k = \arg\max\limits_{h\in\{1,\ldots,K\}} \cos(\boldsymbol{x}_i, \boldsymbol{c}_h), \\ 0, & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, N, \tag{15}$$

$$v_{jq} \;=\; \begin{cases} 1, & \text{if } q = \arg\max\limits_{r\in\{1,\ldots,Q\}} \cos(\boldsymbol{x}_{\cdot j}, \boldsymbol{d}_r), \\ 0, & \text{otherwise,} \end{cases} \qquad j = 1, \ldots J. \tag{16}$$

Although cosine similarity is traditionally used for clustering only units by rows (terms), in SDKM it is applied symmetrically to both terms and documents, relying on the assumption that both reside in high-dimensional sparse spaces where angular similarity captures meaningful structure. Assume non-zero rows of $\boldsymbol{X}$ and the centroid rows of $\overline{\boldsymbol{Y}}$ are compared via cosine, and $\boldsymbol{c}_k$ and $\boldsymbol{d}_q$ are defined as in Eqs. 13 and 14. Then:

(i) For fixed $(\boldsymbol{V}, \overline{\boldsymbol{Y}})$, maximizing $f(\boldsymbol{U}, \boldsymbol{V}, \overline{\boldsymbol{Y}})$ over $\boldsymbol{U}$ reduces to assigning each i to $\arg\max_k \cos(\boldsymbol{x}_i, \boldsymbol{c}_k)$, i.e., Eq. 15.

(ii) For fixed $(\boldsymbol{U}, \overline{\boldsymbol{Y}})$, maximizing over $\boldsymbol{V}$ reduces to assigning each j to $\arg\max_q \cos(\boldsymbol{x}_{\cdot j}, \boldsymbol{d}_q)$, i.e., Eq. 16.

Consequently, SDKM performs block–coordinate ascent on a directional (angle-based) objective that is the co-clustering analog of SKM: the row and column updates are cosine—likelihood assignments in dual spaces coupled through $\overline{\boldsymbol{Y}}$ (since $\boldsymbol{c}_k$ depends on $\boldsymbol{V}$ and $\boldsymbol{d}_q$ on $\boldsymbol{U}$). Equivalently, the assignments coincide with those from two conditionally independent mixtures of von Mises–Fisher (vMF) components (one for rows, one for columns) with common positive concentrations (Hornik & Grün, 2014), because the complete-data log–likelihood is proportional to

$$\sum_{i,k} u_{ik} \cos(\boldsymbol{x}_i, \boldsymbol{c}_k) + \sum_{j,q} v_{jq} \cos(\boldsymbol{x}_{\cdot j}, \boldsymbol{d}_q),$$

up to constants.

For (i), with $(\boldsymbol{V}, \overline{\boldsymbol{Y}})$ fixed, $f = \operatorname{tr}(\boldsymbol{X}'\boldsymbol{U}\overline{\boldsymbol{Y}}\boldsymbol{V}') = \sum_i \boldsymbol{x}_i'(\overline{\boldsymbol{Y}}\boldsymbol{V}')_{k(i)}$, where $k(i)$ is the selected cluster for row i. Since $(\overline{\boldsymbol{Y}}\boldsymbol{V}')_{k\cdot} = \boldsymbol{c}_k$ and argmax is unchanged by positive rescalings, the optimal $k(i)$ maximizes $\boldsymbol{x}_i'\boldsymbol{c}_k$, i.e., $\cos(\boldsymbol{x}_i, \boldsymbol{c}_k)$ after normalizing. Part (ii) is analogous, using $\boldsymbol{d}_q = \boldsymbol{U}\overline{\boldsymbol{y}}_{\cdot q}$. The vMF view follows because maximizing $\sum \cos(\cdot, \cdot)$ equals maximizing a vMF complete-data likelihood with fixed concentrations (monotone reparameterization).

Remark 4 (Interdependence and appropriateness of cosine in co-clustering) Cosine acts on orientations (not magnitudes), which is desirable in sparse high-dimensional text. In SDKM, the dual use of cosine is principled because the centroids $\{\boldsymbol{c}_k\}$ and $\{\boldsymbol{d}_q\}$ are not independent: they are linear images of the common latent block structure $\overline{\boldsymbol{Y}}$ through $\boldsymbol{V}$ and $\boldsymbol{U}$, respectively. Thus, row assignments exploit column information and vice versa via the same bilinear factor $\boldsymbol{U}\overline{\boldsymbol{Y}}\boldsymbol{V}'$, ensuring that the two cosine decisions optimize a single global objective rather than two unrelated ones (Banerjee et al., 2005; Dhillon & Modha, 2001).

The spherical double k-means algorithm proceeds as follows:

Algorithm 1 Spherical double k-means (SDKM)

Require: $\boldsymbol{X} \in \mathbb{R}^{N\times J}$, numbers of clusters (K, Q), tolerance $\varepsilon > 0$, maximum iterations T.
1: Initialize $\boldsymbol{U} \in \{0,1\}^{N\times K}$ and $\boldsymbol{V} \in \{0,1\}^{J\times Q}$ to be row-stochastic (e.g., random partitions).
2: Compute $\overline{\boldsymbol{Y}}$ by Eq. 11; set $t \leftarrow 0$ and $f^{(0)} \leftarrow f(\boldsymbol{U}, \boldsymbol{V}, \overline{\boldsymbol{Y}})$ using Eq. 12.
3: **repeat**
4: **Update** $\boldsymbol{U}$ by the cosine assignments in Eq. 15 using centroids $\boldsymbol{c}_k$ from Eq. 13.
5: **Update** $\overline{\boldsymbol{Y}}$ by Eq. 11.
6: **Update** $\boldsymbol{V}$ by the cosine assignments in Eq. 16 using centroids $\boldsymbol{d}_q$ from Eq. 14.
7: **Update** $\overline{\boldsymbol{Y}}$ again by Eq. 11.
8: **compute** $f^{(t)}$ by Eq. 12; $t \leftarrow t+1$.
9: **until** $\mid f^{(t)} - f^{(t-1)} \mid < \varepsilon$ **or** $t \geq T$

As a stopping rule, we monitor the normalized objective in Eq. 12. Let $f^{(t)} = f(\boldsymbol{U}^{(t)}, \boldsymbol{V}^{(t)}, \overline{\boldsymbol{Y}}^{(t)})$. We stop when $\mid f^{(t)} - f^{(t-1)} \mid < \varepsilon$ (with $\varepsilon \in [10^{-9}, 10^{-6}]$ in our experiments) or when a maximum number of iterations T is reached (T=100 in our experiments). The iteration index t is increased once at the end of each outer loop.

Here k and q index clusters, and the assignments in Eqs. 15 and 16 choose, respectively, the row and column cluster that maximizes the cosine with the current directional centroids $\boldsymbol{c}_k$ and $\boldsymbol{d}_q$.

The above algorithm guarantees that the objective function's value increases monotonically at each iteration.

Indeed, it can be shown that for any $t \geq 0$:

$$f(\boldsymbol{U}^{(t)}, \overline{\boldsymbol{Y}}^{(t)}, \boldsymbol{V}^{(t)}) \leq f(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+1)}, \boldsymbol{V}^{(t+1)}).$$

Letting $p_k^{(t)}$ denote the centroid of the generic cluster k of the units, i.e., rows, at step t of the algorithm, with $1 \leq k \leq K$, and $\pi_k^{(t)}$ the corresponding cluster, the previous inequality is demonstrated in Appendix 1. Since the underlying optimization problem is NP-hard, the algorithm cannot guarantee convergence to a global optimum. This is a well-known limitation in clustering methods such as SKM and DKM. To mitigate this issue in practice, we suggest running the algorithm from multiple random initializations (or random starts), as is common in similar settings.

3.1 Normalization Issue

In spherical clustering, it is standard to constrain centroids to lie on the unit hypersphere in order to meaningfully interpret distances in terms of cosine similarity. This requires that the centroid vectors, typically denoted as rows of the matrix $\overline{\boldsymbol{Y}}$, be normalized to unit norm

(Dhillon & Modha, 2001). Such normalization ensures that the inner product between data points and centroids directly corresponds to cosine similarity, which is particularly appropriate for high-dimensional sparse data like text (Banerjee et al., 2005).

However, in a co-clustering framework, where clustering is performed simultaneously on both the rows and the columns of the data matrix, the normalization of the centroid matrix becomes more nuanced. Since $\overline{\boldsymbol{Y}}$ encodes the interactions between row and column clusters, enforcing unit-norm constraints on both rows and columns simultaneously is generally not feasible. Such a constraint would overdetermine the solution space and potentially distort the optimization objective, due to the inherent interdependence between the two clustering dimensions. This phenomenon is well-recognized in the co-clustering literature, where normalization is typically applied in a single dimension to preserve flexibility and ensure model stability (Dhillon, 2001; Kluger et al., 2003).

In co-clustering (or biclustering) approaches, the centroid matrix $\overline{\boldsymbol{Y}}$ represents the latent structure of associations between row clusters (e.g., terms) and column clusters (e.g., documents). Each entry in $\overline{\boldsymbol{Y}}$ reflects the strength of the relationship between a specific term cluster and a document cluster.

When we normalize the rows of $\overline{\boldsymbol{Y}}$, we enforce unit-norm constraints on the cluster representations of terms, which is desirable when cosine similarity is used as a distance metric. However, applying the same constraint to the columns of $\overline{\boldsymbol{Y}}$ introduces a second layer of normalization that interferes with the optimization process. Specifically, because $\overline{\boldsymbol{Y}}$ is learned as a solution to a constrained optimization problem, normalizing both rows and columns simultaneously imposes overly restrictive conditions. This can excessively reduce the space of feasible solutions and/or make the optimization problem ill-posed or numerically unstable (Lee & Seung, 1999). In addition, it can also disrupt the balance between the row and column clustering objectives.

Crucially, the two clustering dimensions are not independent: modifying the norm of the row vectors in $\overline{\boldsymbol{Y}}$ affects the interpretation and optimization of the column clusters, and vice versa. This interdependence is acknowledged in several foundational works on co-clustering and matrix factorization-based approaches, where normalization is typically applied to only one dimension in order to preserve the stability and interpretability of the model.

To address this, we choose to normalize the centroid matrix $\overline{\boldsymbol{Y}}$ by rows only. This decision is motivated by the nature of textual data, where terms (represented by the rows in the input matrix $\boldsymbol{X}$) tend to exhibit greater variability and noise compared to documents (the columns). Normalizing by rows ensures that each term-cluster centroid has unit norm, which aligns with the geometric assumptions of spherical clustering and improves robustness to outliers and frequency disparities in the vocabulary (Zhong & Ghosh, 2005).

Importantly, although we do not explicitly normalize the columns of $\overline{\boldsymbol{Y}}$, we observe empirically that their norms tend to remain close to one throughout the optimization. This suggests that the absence of column normalization does not introduce significant bias or instability in practice, and the cosine-based interpretation of similarity is preserved to a reasonable extent in both dimensions. We further validate this empirical behavior in Sects. 5 and 6, where we monitor the column norms of $\overline{\boldsymbol{Y}}$. The analysis confirms their proximity to one, suggesting that explicit column normalization is not critical for stability or interpretability. Note that the cosine-based assignments in Eqs. 15 and 16 are invariant to any positive rescaling of rows/columns of $\boldsymbol{X}$ and to diagonal rescalings of $\overline{\boldsymbol{Y}}$ (the norms cancel inside cos), and the normalized objective in Eq. 12 is homogeneous of degree zero. Hence, enforcing unit-norm rows of $\overline{\boldsymbol{Y}}$ is sufficient for identifiability, while leaving columns unnormalized does not bias assignments in either dimension.

4 Computational Details

Each start draws $\boldsymbol{U} \in 0, 1^{N\times K}$ and $\boldsymbol{V} \in 0, 1^{J\times Q}$ as random hard partitions with all clusters non-empty (we then apply a light balancing move after each assignment step to prevent empty clusters), and centroids are computed as block means on the preprocessed matrix $\boldsymbol{X}_s$ and row-normalized to unit length, i.e.,

$$\overline{\boldsymbol{Y}} = \text{row-norm}\big((\boldsymbol{U}'\boldsymbol{U})^{-1}\boldsymbol{U}'\boldsymbol{X}_s, \boldsymbol{V}(\boldsymbol{V}'\boldsymbol{V})^{-1}\big).$$

We use a multi-start scheme and retain the solution with the largest value of the normalized objective in Eq. 12; the data-driven selection of the number of starts and its effect on local optima is given in Sect. 5 (Table 4), and in all reported experiments, we set random starts `Rndst` $= 20$ for each (K, Q). The stopping rule and iteration cap follow the specification given after Algorithm 1; empirically, runs stabilized well before the cap in both simulations and the application. For the application, we fixed a pseudorandom seed, and after convergence, we reorder clusters by decreasing size in $\boldsymbol{U}$ and $\boldsymbol{V}$ to stabilize labeling (this does not affect the objective).

4.1 Empirical Running Times and Scaling Behavior

To complement the algorithmic details, we report wall–clock times for a full grid over (K, Q) on a representative sparse TF–IDF text matrix ($N = 1278$ terms, $J = 2834$ documents; see

Table 1 SDKM runtime summary by K (aggregating over Q)

K	Median	IQR	Min	Max	Slope$_{/Q}$	R^2
2	537	138	193	716	18.30	0.82
3	587	149	301	810	15.60	0.66
4	528	142	196	710	20.30	0.88
5	531	189	242	720	19.00	0.87
6	608	122	282	805	19.30	0.84
7	630	220	327	825	22.40	0.95
8	657	193	404	921	18.50	0.82
9	697	133	363	950	18.90	0.77
10	740	125	306	887	17.80	0.62
11	680	162	308	883	21.00	0.89
12	739	145	381	876	17.90	0.81
13	711	151	371	911	19.30	0.81
14	461	63.40	416	742	-6.41	0.22
15	456	51.80	385	496	-0.29	0.00
16	431	43.60	334	496	2.01	0.11
17	475	74.10	314	511	7.33	0.71
18	449	22.40	356	499	2.01	0.17
19	446	55.30	335	507	4.30	0.34
20	447	56.10	361	502	3.43	0.36
21	466	43	332	542	3.22	0.19
22	470	34.20	322	546	4.85	0.45

"Slope$_{/Q}$" is the least–squares slope of runtime on Q for fixed K; R^2 is the corresponding coefficient of determination

Table 2 SDKM runtime summary by Q (aggregating over K)

Q	Median	IQR	Min	Max	Slope$_{/K}$	R^2
2	332	56.80	193	479	5.65	0.24
3	427	96.60	306	563	2.10	0.03
4	433	102	340	658	0.25	0
5	478	83.30	392	742	0.39	0
6	478	163	411	645	–4	0.09
7	510	157	406	711	–7.27	0.22
8	477	142	401	752	–6.55	0.18
9	509	142	411	749	–4.10	0.08
10	512	164	405	697	–7.53	0.26
11	531	188	386	796	–8.38	0.20
12	546	213	405	887	–9.69	0.20
13	560	193	406	777	–9.92	0.29
14	527	244	399	814	–10.80	0.23
15	582	258	420	834	–12.40	0.33
16	620	273	382	792	–15.40	0.45
17	624	267	434	921	–17.60	0.49
18	625	251	416	840	–14.80	0.42
19	626	274	418	864	–14.60	0.39
20	649	309	395	950	–16.60	0.33
21	698	343	414	890	–18.80	0.46
22	710	287	413	911	–16.70	0.40

"Slope$_{/K}$" is the least–squares slope of runtime on K for fixed Q; R^2 is the corresponding coefficient of determination

Sect. 6 for the application pipeline). The best normalized objective across starts was retained. The complete per–cell timing matrix appears in the supplementary material. For interpretability we also summarize, across one dimension at a time, the median and interquartile range (IQR) of runtimes together with the least–squares slope of time versus the other cluster count (seconds per added cluster) and its coefficient of determination R^2.

From these summaries we get that for small/moderate K the median runtime grows almost linearly in Q (Table 1, Slope$_{/Q} \approx$ 18–22 seconds per cluster with large R^2), whereas the dependence on K at fixed Q is weaker and often non–monotone (Table 2, Slope$_{/K}$ small or negative with low R^2), reflecting that faster convergence at larger K can offset the extra per–iteration work. For sparse, high–dimensional text matrices, runtime is chiefly driven by the cluster count on the side that enters dense centroid updates, consistent with the per–iteration cost.

5 Cluster Validity and Simulation Study

The simulation study has been developed in order to test the model and algorithm's performance. We implemented a data generation procedure in order to generate datasets whose clustering structure follows an SDKM model. The generated datasets are then used to focus on different task: (i) development of the pseudo-F index, proposed by Vichi (2015), to detect

the true number of clusters of units K and the true number of clusters of variables Q; (ii) implementation of different scenarios (corresponding to different error levels) in order to assess model performance in terms of true partitions recovery, measured with the Adjusted Rand Index (ARI; Hubert and Arabie, 1985), and centroids matrix recovery, measured with Rooted Mean Squared Error (RMSE) and its Normalized versions, i.e., Normalized Rooted Mean Squared Error (NRMSE). Clearly, there exist several normalization techniques: we decided to implement NRMSE1, where the RMSE is normalized by dividing it by the range of the true centroids matrix; NRMSE2, instead, is obtained by dividing RMSE by the norm of the centered centroids matrix. The simulation study has been developed in order to test the model and algorithm's performance on general high-dimensional data (textual corpora are one instance). We implemented a data generation procedure to create matrices whose clustering structure follows an SDKM model, regardless of the application domain.

5.1 Data Generation Mechanism

We implemented a MATLAB procedure that generates a matrix with an SDKM-type clustering structure. The protocol adapts the simulation schemes used for DKM in Rocci and Vichi (2008) to the spherical setting, so it is not fully original but tailored here to SDKM.

The term-document data matrix $\boldsymbol{X}$ is generated as follows:

$$\boldsymbol{X} = \boldsymbol{U}\overline{\boldsymbol{Y}}\boldsymbol{V}' + \boldsymbol{E},$$

where $\mathbf{U}$ is the $(N \times K)$ membership matrix of units; $\mathbf{V}$ is the $(J \times Q)$ membership matrix of variables; $\overline{\boldsymbol{Y}}$ is the $(K \times Q)$ matrix of centroids; $\mathbf{E}$ is the $(N \times J)$ matrix of errors, generated from a normal distribution centered in $\mathbf{0}$. The membership matrices have been generated by randomly permuting the rows of an initial matrix, where the first rows form an identity matrix and the remaining rows contain randomized binary vectors with a 1 in the first column, ensuring variability while maintaining a structured pattern; the matrix of centroids has been generated in such a way centroids are equidistant: a regular 10-dimensional simplex has been generated; then, a multidimensional scaling has been applied; finally, the centroid matrix is obtained by multiplying the largest real eigenvectors and the square root of the largest real eigenvalues. This procedure allows to have centroids which are equidistant.

In the generation mechanism, we introduced two sources of error to simulate heterogeneity in the data: a centroid error, denoted by $\epsilon_{\text{centroid}}$, which adds variability to the cluster centroids and mimics within-cluster noise, and a cluster error, denoted by $\epsilon_{\text{cluster}}$, which introduces variability between clusters and affects the distinctness of the cluster separation.

5.2 Selection of the Number of Clusters

To determine the appropriate numbers of clusters of terms and of documents, K and Q, respectively, we employed the pseudo-F index proposed by Rocci and Vichi (2008). This index, which is based on the criterion proposed by Caliński and Harabasz (1974), has previously been utilized for the DKM algorithm. To the best of our knowledge, while pseudo-F type criteria have been used in Euclidean k-means settings, their explicit use in spherical k-means and, more generally, in spherical co-clustering methods such as SDKM has not been discussed. Here, we adapt the same between/within deviance rationale to the spherical setting by computing the index on the normalized representation adopted by SDKM, so that the resulting criterion is coherent with the geometry underlying the clustering objective.

$$pF_{dk} = \frac{\left\| \boldsymbol{H}_U \boldsymbol{X} \boldsymbol{H}_V - (1/NJ)\mathbf{1}_N \mathbf{1}_N' \right\|^2 / (KQ - 1)}{\left\| \boldsymbol{X} - \boldsymbol{H}_U \boldsymbol{X} \boldsymbol{H}_V \right\|^2 / (NJ - KQ)}. \tag{17}$$

In this equation, $\boldsymbol{H}_U$ and $\boldsymbol{H}_V$ are projection matrices that map the data onto the subspaces defined by the unit and feature clusters, respectively, and $\mathbf{1}_N$ is a vector of ones of length N. The numerator represents the between-cluster deviance, while the denominator accounts for the within-cluster deviance, with each term normalized by its corresponding degrees of freedom.

Although the pseudo-F index is written in terms of squared Frobenius norms, it admits a direct interpretation in spherical terms. Indeed, when document/term profiles are ℓ_2-normalized (as in SDKM), for any unit vectors $\boldsymbol{x}$ and $\boldsymbol{c}_k$ we have $\|\boldsymbol{x} - \boldsymbol{c}_k\|_2^2 = \boldsymbol{x}'\boldsymbol{x} + \boldsymbol{c}_k'\boldsymbol{c}_k - 2\boldsymbol{x}'\boldsymbol{c}_k = 2\{1 - \boldsymbol{x}'\boldsymbol{c}_k\} = 2\{1 - \cos(\boldsymbol{x}, \boldsymbol{c}_k)\} = 2\,d(\boldsymbol{x}, \boldsymbol{c}_k)$, where $\cos(\boldsymbol{x}, \boldsymbol{c}_k)$ and $d(\boldsymbol{x}, \boldsymbol{c}_k) = 1 - \cos(\boldsymbol{x}, \boldsymbol{c}_k)$ are the cosine similarity/dissimilarity defined in Eqs. 3–4, and $\theta = \arccos(\cos(\boldsymbol{x}, \boldsymbol{c}_k))$ is the corresponding angular distance. Therefore, the within-cluster deviance appearing in the denominator is proportional to an aggregated cosine/angle-based dispersion, while the between-cluster deviance in the numerator captures separation in the same spherical geometry. In this sense, maximizing pseudo-F favors solutions with small average within-cluster angular dispersion and large between-cluster angular separation, consistently with the SDKM objective. Accordingly, the optimal values of K and Q are identified by locating a (local) maximum of the pF_{dk} index across SDKM solutions fitted over a grid of (K, Q) values. As noted by Rocci and Vichi (2008), this index is effective in scenarios where the data exhibit a well-defined cluster structure.

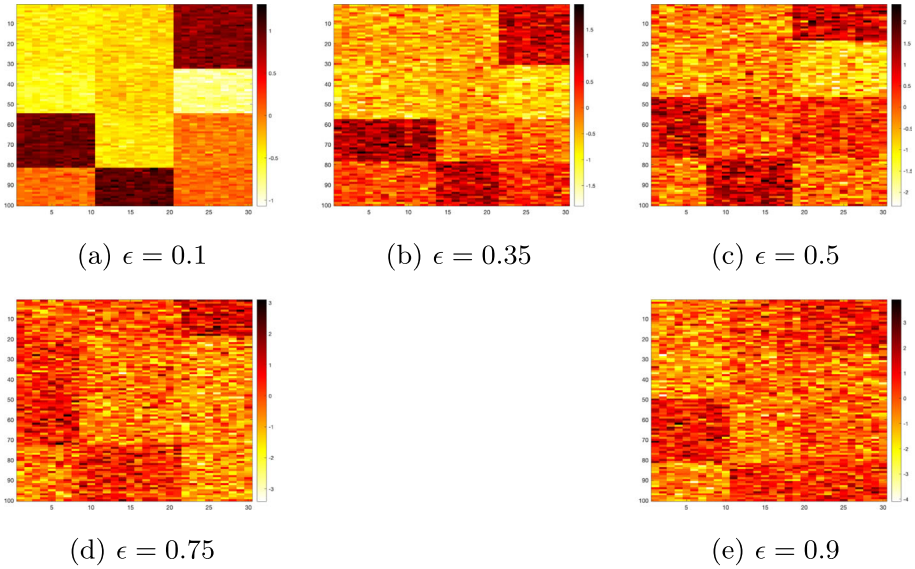
5.3 Pseudo-F Index Simulation Study

We conducted a simulation study to evaluate the effectiveness of the pseudo-F index in selecting the correct number of clusters K and Q under varying levels of error. In our simulations, we generated synthetic data with known cluster structures. The term-document data matrix $\boldsymbol{X}$ was constructed based on the true cluster configurations, with $K_{\text{true}} = 4$ clusters for units and $Q_{\text{true}} = 3$ clusters for variables.

For simplicity, we set both error terms to the same value in each simulation run ($\epsilon_{\text{centroid}} = \epsilon_{\text{cluster}} = \epsilon$), where ϵ takes on values of 0.1, 0.35, 0.5, 0.75, and 0.9. The data at the different error levels are displayed in Fig. 1. By varying ϵ simultaneously for both error sources, we assessed the combined impact of increasing noise levels on the ability of the pseudo-F index to correctly identify the true number of clusters.

We performed 100 runs, varying K and Q from 2 to 5. For each run, we calculated the pseudo-F index for each combination of K and Q and recorded the number of times each combination resulted in the highest pseudo-F value. Since in Eq. 17, $KQ - 1 = 0$ when $(K, Q) = (1, 1)$, the pseudo-F index is not defined for the trivial null configuration, and we evaluate it for $K, Q \geq 2$.

The results, summarized in Table 3, indicate that at a low error level ($\epsilon = 0.1$), the pseudo-F index correctly identified the true number of clusters ($K = 4$, $Q = 3$) in 100% of the runs. However, as the error level increased, the pseudo-F index increasingly favored solutions with fewer clusters. For instance, at $\epsilon = 0.5$, the correct combination was selected in only 13% of the runs, while combinations with $K = 2$ or $K = 3$ and $Q = 2$ became more prevalent.



(a) $\epsilon = 0.1$ (b) $\epsilon = 0.35$ (c) $\epsilon = 0.5$

(d) $\epsilon = 0.75$ (e) $\epsilon = 0.9$

Fig. 1 Heatmaps of the data at different error levels ϵ

Table 3 Number of times the true combination ($K = 4$, $Q = 3$) had the highest pseudo-F index across 100 runs for different error levels ϵ. Bold entries indicate the number of times the true combination is catched

		Q			
ϵ	K	2	3	4	5
0.1	2	0	0	0	0
	3	0	0	0	0
	4	0	**100**	0	0
	5	0	0	0	0
0.35	2	0	0	0	0
	3	15	0	0	0
	4	0	**85**	0	0
	5	0	0	0	0
0.5	2	20	0	0	0
	3	66	0	0	0
	4	1	**13**	0	0
	5	0	0	0	0
0.75	2	58	1	0	0
	3	37	2	0	0
	4	0	**2**	0	0
	5	0	0	0	0
0.9	2	70	0	0	0
	3	30	0	0	0
	4	0	**0**	0	0
	5	0	0	0	0

At the highest error level ($\epsilon = 0.9$), the pseudo-F index did not select the true number of clusters in any run, instead favoring the combination ($K = 2$, $Q = 2$) in 70% of the runs. These results indicate that the pseudo-F index is robust in identifying the correct number of clusters in data with a clear cluster structure, but tends to underestimate the number of clusters when the data become noisier. For this reason, we use the index as a screening rule to identify plausible (K, Q) values and complement it with substantive interpretability (and, when relevant, stability considerations) in the real-data analyses.

It must be noticed that performing 100 simulation runs on a MacBook equipped with an Apple M1 chip and 8 GB of RAM, took on average approximately 87 seconds for the total running time, meaning each run took less than a second on average. This indicates a substantial reduction in computation time compared to traditional SKM algorithms, which are typically computationally intensive (Duwairi & Abu-Rahmeh, 2015; Kim et al., 2020). The improved efficiency is attributed to our use of matrix operations in the implementation, which streamlines calculations and minimizes processing overhead.

5.4 Selection of Random Starts

Since the partitioning problem is known to be NP-hard, the new methodology is not guaranteed to obtain a global solution. Therefore, this section is devoted to discuss the selection of random starts.

First, model 5.1 is used to generate a term-document data matrix $\mathbf{X}$ with a high error level ($\epsilon_{\text{centroid}} = \epsilon_{\text{cluster}} = \epsilon = 1.5$): therefore, the true partitions $\mathbf{U}$, $\mathbf{V}$, the true centroids $\overline{\mathbf{Y}}$ and the true value of the objective function are known.

Then, the algorithm is run 100 times by letting the number of random starts take values in [1, 5, 10, 20, 30, 40, 50, 70, 100]. For each random start, the algorithm is therefore run 100 times, and the values of the objective function are stored. At the end, for each random start, we obtain a distribution of 100 values of the objective function. Clearly, it is possible to compare those values with the true one. Whenever the algorithm returns a value of the objective function lower than the true one, the algorithm is trapped in a local maximum.

It is advised to select the number of random starts in such a way that a local maximum never occurs.

The percentage of local maxima occurrences for each random start is reported in Table 4. The percentage of local maxima always decreases with the increase of the RndStarts until reaching 0, when the number of random starts is set equal to 20. Thus, the selected number of random starts for the whole simulation study was set to RndStarts = 20.

From Table 4 it can be observed that with 1 random start of the algorithm and high error, the new methodology performs bad, identifying the optimal solution in only 29% of cases. It is worth noticing that with 10 random starts, the algorithm identifies the global optimal solution in 91% of cases, and with 20 random starts, the algorithm was never trapped in local maxima.

Table 4 Local maxima occurrences (%) with high error

Random start	1	5	10	20	30	40	50	70	100
% of local maxima	71	23	9	0	0	0	0	0	0

5.5 Performance in Terms of True Partitions and Centroids Recovery

We conducted a simulation study to evaluate the effectiveness of the algorithm in recovering the correct true partitions **U** and **V** and the centroids matrix $\overline{\mathbf{Y}}$ under varying levels of error. In our simulations, we generated synthetic data with known cluster structures by using the model 5.1. The term-document data matrix $\boldsymbol{X}$ was constructed based on the true cluster configurations, with $K_{\text{true}} = 3$ clusters for units and $Q_{\text{true}} = 2$ clusters for variables.

For simplicity, we set both error terms to the same value in each simulation run ($\epsilon_{\text{centroid}} = \epsilon_{\text{cluster}} = \epsilon$), where ϵ takes on values of 0.1, 0.35, 0.5, 0.75, 0.9, 1.1, 1.35, 1.5, 1.75, and 2. By varying ϵ simultaneously for both error sources, we assessed the combined impact of increasing noise levels on the ability of the algorithm to recover the underlying partitions and the centroids matrix.

For each error level, we performed 500 runs, for a total of 5000 samples. For each run, and for each error level, we calculated the ARI index for both **U** and **V** and the Root Mean Square Error (RMSE) and its normalized versions, NRMSE1 and NRMSE2. Obviously, the higher the ARI, the better the similarity between the obtained and true partitions; the lower the RMSE, NRMSE1, NRMSE2, the closer the obtained and the true centroids matrices. The normalized versions of the RMSE range are [0, 1].

The results, summarized in Table 5, indicate that when the error level increases, the performance gets worse, as expected. More in details, at a low error level ($\epsilon = 0.1$), the true partitions are perfectly recovered, as well as the centroid matrix. By increasing the error level, the ARI indices decreases and the RMSE, NRMSE1, NRMSE2 indices increase. For instance, with error level ϵ equal to 1.35, the summary statistics of ARI indices are about 0.7 and the average NRMSEs are close to 0.08. It is worth noticing that the median value of ARI for **V** remains equal to 1 by increasing the error level until 1.35, meaning that half of the runs completely recovers the true partition. Instead, when the error level reaches 1.75, the median ARI for **V** rapidly declines toward 0.522, meaning that in half of the runs, the true partition is recovered only partially. When the error level is set to 2, then the partitions are not recovered: the obtained ARI indices are slightly above 0, a value that is reached only when the comparing partition is a random one (Hubert & Arabie, 1985). For what concerns the centroids matrix recovery, NRMSEs statistics are all above 0.1.

To summarize, we can conclude that with an increasing level of error, the recovery of the true partitions become a really hard task, since the error terms mask the true underlying partition. On the contrary, with an increasing level of error, the recovery of the centroid matrix is still quite feasible, as the summary values of the NRMSEs never reach or become close to their maximum, i.e., 1.

5.5.1 Column Norms and Approximate Normalization

As discussed in Sect. 3.1, the rows of the centroid matrix $\overline{\boldsymbol{Y}}$ are explicitly normalized to unit norm, while the columns are not, due to the inherent constraints of co-clustering. Although exact normalization in both dimensions is not feasible, we empirically assess the extent to which column norms deviate from 1.

Table 6 reports the norms of the $Q = 2$ column vectors of $\overline{\boldsymbol{Y}}$ across the simulation framework. The results show that the column norms remain consistently close to one, providing empirical support for the stability of the method and the approximate validity of the spherical assumption along the column dimension as well.

Table 5 Summary statistics to evaluate algorithm's performance under low and high level of error

index	statistic	Error level ϵ									
		0.10	0.35	0.50	0.75	0.90	1.10	1.35	1.50	1.75	2.00
ARI for U	Mean	1	1	1	0.974	0.907	0.794	0.675	0.587	0.480	0.354
	Median	1	1	1	1	0.976	0.877	0.682	0.567	0.467	0.360
ARI for V	Mean	1	1	1	0.984	0.933	0.840	0.717	0.649	0.507	0.332
	Median	1	1	1	1	1	1	1	0.866	0.522	0.192
RMSE	Mean	0.003	0.010	0.015	0.03	0.044	0.082	0.132	0.167	0.225	0.314
	Median	0.003	0.009	0.015	0.02	0.031	0.047	0.070	0.098	0.159	0.260
NRMSE1	Mean	0.001	0.005	0.008	0.016	0.025	0.045	0.074	0.094	0.129	0.180
	Median	0.001	0.005	0.008	0.012	0.017	0.025	0.040	0.056	0.088	0.145
NRMSE2	Mean	0.002	0.006	0.009	0.018	0.027	0.050	0.082	0.104	0.143	0.198
	Median	0.002	0.006	0.008	0.013	0.018	0.027	0.043	0.061	0.099	0.162

Table 6 Column norms of obtained centroids under varying error level ϵ

q	Error level ϵ 0.10	0.35	0.50	0.75	0.90	1.10	1.35	1.50	1.75	2.00
q=1	1.193	1.201	1.190	1.175	0.938	1.117	1.239	1.161	1.192	1.189
q=2	1.256	1.248	1.258	1.273	1.456	1.323	1.210	1.286	1.257	1.259

6 U.S. Presidential Inaugural Addresses Application

We applied the SDKM method to a dataset from the R package quanteda, containing U.S. presidential inaugural addresses, ranging from those delivered by George Washington in 1789 to Joe Biden in 2021. Our objective is to identify shared topics among speeches given by different presidents. Using SDKM allows for the simultaneous clustering of both documents and terms, providing a comprehensive view of the evolving presidential discourse. The initial corpus comprises 59 documents, with 151,536 tokens, 9442 types, and 4218 hapaxes. On average, each document contains 770.4 types and 2568 tokens. We calculated the type/token ratio, 0.0623, and the hapax percentage, 0.4467. These values suggest a high lexical variety, justifying the computational analysis of the corpus.

In our analysis, data cleaning and preprocessing were crucial. This involved tokenizing texts and removing extraneous elements such as numbers and punctuation. Lemmatization was applied to reduce each word to its base form. We built a document-term matrix, excluding stop-words and trimming infrequent words with a frequency of 11 or less. These steps were necessary to focus the analysis on core topics articulated by the presidents. We employed the TF-IDF weighting scheme, which assesses a word's significance within documents relative to the entire corpus (Liang & Niu, 2022; Manning, 2009). This approach helps mitigate length-induced bias in document analysis by focusing on term importance rather than sheer frequency. Specifically, TF-IDF measures how important a term is within a document relative to a corpus, and it is calculated as:

$$x_{ij} = \frac{n_{ij}}{n_{\cdot j}} \log_{10} \frac{M}{m},$$

where n_{ij} is the raw number of occurrences, M is the number of documents and m is the number of documents that include the term. The final term-document data matrix $\boldsymbol{X}$, comprising $N = 1206$ terms and $J = 59$ documents, demonstrated a sparsity of 67.73%

6.1 Spherical Double K-Means

To determine the best number of clusters for both terms and documents, we used the Pseudo-F index with K and Q ranging from 2 to 10. Table 7 presents these Pseudo-F values, highlighting the clustering dynamics.

The highest Pseudo-F value occurs at $K = 2$ and $Q = 2$, suggesting that the optimal number of clusters for both words and documents is 2. However, as discussed in Sect. 5.3, the Pseudo-F index can tend to underestimate the number of clusters. The second-highest Pseudo-F value occurs at $K = 3$ and $Q = 2$. The observed patterns in the clusters indicate that the choice of $K = 3$ and $Q = 2$ produces clusters with greater interpretability, a crucial

Table 7 Pseudo-F values for combinations of K and Q. Bold entries indicate the highest index values

$K \backslash Q$	2	3	4	5	6	7	8	9	10
2	**1199.5**	675.7	655.6	410.5	452.4	236.1	236.2	286.9	212.9
3	**707.3**	593.9	351.9	241.8	268.8	190.3	162.9	183.6	181.7
4	536.6	417.9	344.4	278.1	167.5	168.0	140.4	131.2	103.2
5	439.7	294.0	258.0	166.1	187.6	130.3	91.7	88.7	131.2
6	358.8	257.0	193.4	150.4	153.7	142.2	99.7	62.9	75.5
7	326.8	190.5	172.1	134.7	90.8	91.8	91.0	59.2	57.1
8	273.3	192.0	162.5	126.2	139.1	84.1	76.2	76.5	67.6
9	247.0	167.4	116.4	107.3	91.6	73.3	64.1	64.6	50.8
10	208.5	165.2	127.0	98.1	82.1	62.8	70.4	46.0	50.8

factor in the effective analysis of clusters as fully discussed in Fraley and Raftery (1998). In addition, to further investigate this aspect, we apply our methodology with $K = 10$ and $Q = 10$, and we plot the clustergram of the centroid matrix. The clustergram in Fig. 2, which shows the hierarchical clustering of the rows and columns of the centroids matrix through a heatmap and two dendrograms, reveals $K = 3$ distinct clusters of words and $Q = 2$ distinct clusters of documents. The dendrograms are obtained by using one minus the sample correlation between points as a distance metric for both rows and columns, and single linkage and Ward linkage as hierarchical clustering methods, for rows and columns, respectively. In addition, the same clustergram procedure is applied to a sample of 150 rows of the theoretical data matrix reconstructed via: $\boldsymbol{X}_t = \boldsymbol{U}\overline{\boldsymbol{Y}}\boldsymbol{V}'$. Figure 3 shows again $K = 3$ distinct clusters of words and $Q = 2$ distinct clusters of documents.

Therefore, $K = 3$ and $Q = 2$ were our final choice as the number of clusters for terms and documents, respectively. For each (K, Q) candidate in Table 7, we ran `Rndst` $= 20$

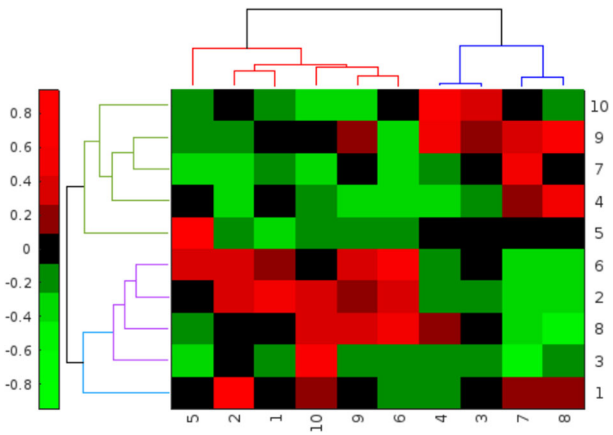


Fig. 2 Clustergram of 10×10 centroids matrix

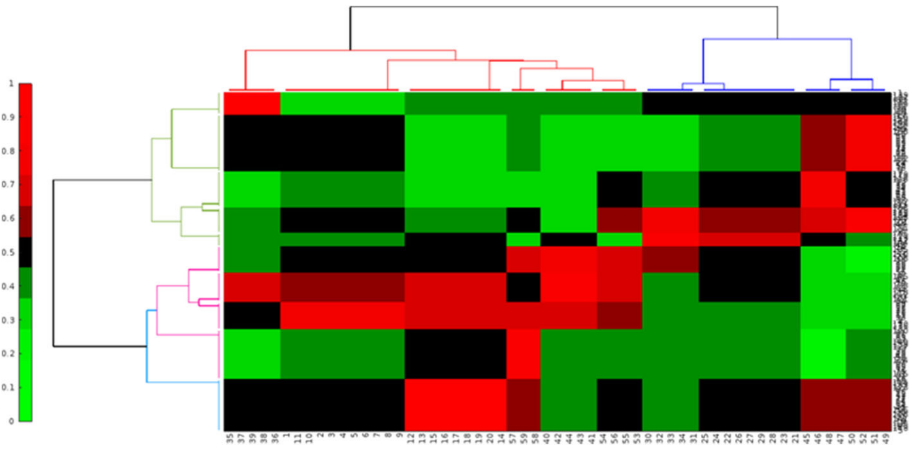


Fig. 3 Clustergram of the theoretical data matrix narrowed down to 150 sampled rows

random starts and retained the solution with the highest value of Eq. 12. In this corpus, we used standardization `'off'` (TF–IDF already controls for document length), a maximum of $T = 100$ iterations, and tolerance $\varepsilon = 10^{-9}$. All runs converged well before the cap.

The analysis provides insights into the evolution of themes in U.S. presidential inaugural addresses. The three clusters of words contain 457 (37.9%), 420 (34.9%), and 328 (27.2%) words, while the two groups of documents contain 31 and 28 documents. The centroids, as described in Sect. 3.1, have row norms exactly equal to 1, while the column norms are 0.869 and 1.498. Although the exact unit norm is not achieved for the columns, these values indicate a reasonable closeness to normalization. This approximate normalization is sufficient to preserve the interpretability of the cosine similarity measure and does not appear to compromise the stability or quality of the clustering results in practice.

6.1.1 Document Distribution and Historical Context

Figure 4 presents the document distribution across SDKM's two clusters, along with key events in U.S. history. Cluster 1 spans speeches from 1789 to 1861, reflecting the early nation-building era, while Cluster 2 covers addresses from 1865 onward, aligning with the modern evolution of presidential rhetoric.

The division between clusters aligns closely with significant historical periods. Lincoln's 1861 speech, placed in cluster 1, focuses on the legal implications of secession and efforts to preserve the union. In contrast, his 1865 speech, included in cluster 2, emphasizes reconciliation and future aspirations, aligning with themes of national healing more typical of the recent years cluster. Interestingly, Calvin Coolidge (1925) and Herbert Hoover (1929) are later exceptions in Cluster 1. Their speeches likely focus on governmental and economic policies prior to the great depression, which aligns more closely with the themes prevalent in cluster 1.

6.1.2 Word Clusters and *Thematic Interpretation*

Figure 5 displays the top 30 terms in each of the three word clusters. Cluster 1, that we labeled "**American Dream**," features words like "america," "freedom," "democracy," and "dream,"

pointing to aspirational rhetoric and national ideals. Cluster 2 has been labeled "**Law and Order**," it centers on terms like "union," "constitution," "state," and "government," underscoring legal and institutional frameworks. We called Cluster 3, "**Politics, Economics, and Secession**." It includes words such as "congress," "law," "business," and "policy," reflecting deeper political processes and economic discussions.

In the simplex plot illustrated in Fig. 6, points represent presidential speeches, colored according to their cluster membership. The blue points, corresponding to the "older" presidents cluster, are mainly distributed between the Politics/Economics/Secession vertex and the Law and Order vertex. This suggests that their speeches emphasize themes related to politics, economics, and security, reflecting a strong focus on order and institutional structures. Some of these blue points lie near the center of the triangle, indicating that certain speeches combine the three themes more evenly, without a clear dominance of one. The green points, associated with the "modern" presidents cluster, are concentrated around the American Dream vertex, with some positioned between the American Dream and Politics/Economics/Secession ver-

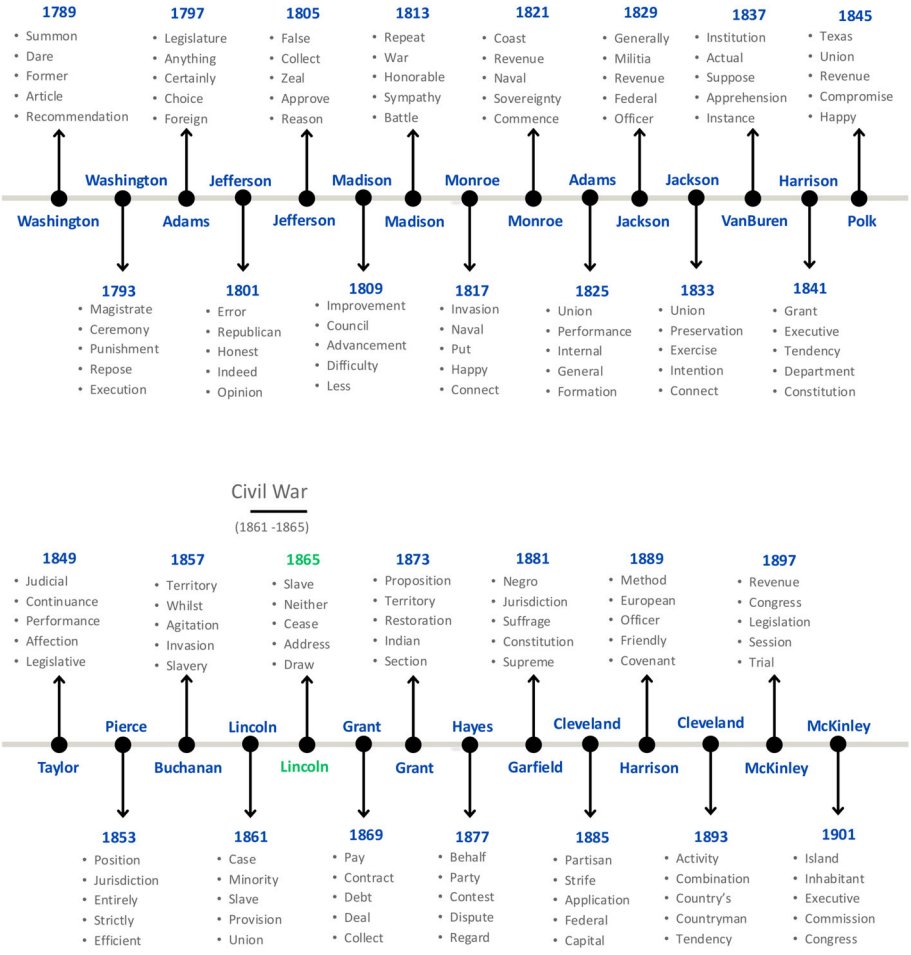


Fig. 4 Historical timeline of the US presidents, colored by cluster of documents and their 5 most frequent words (TF-IDF normalized)

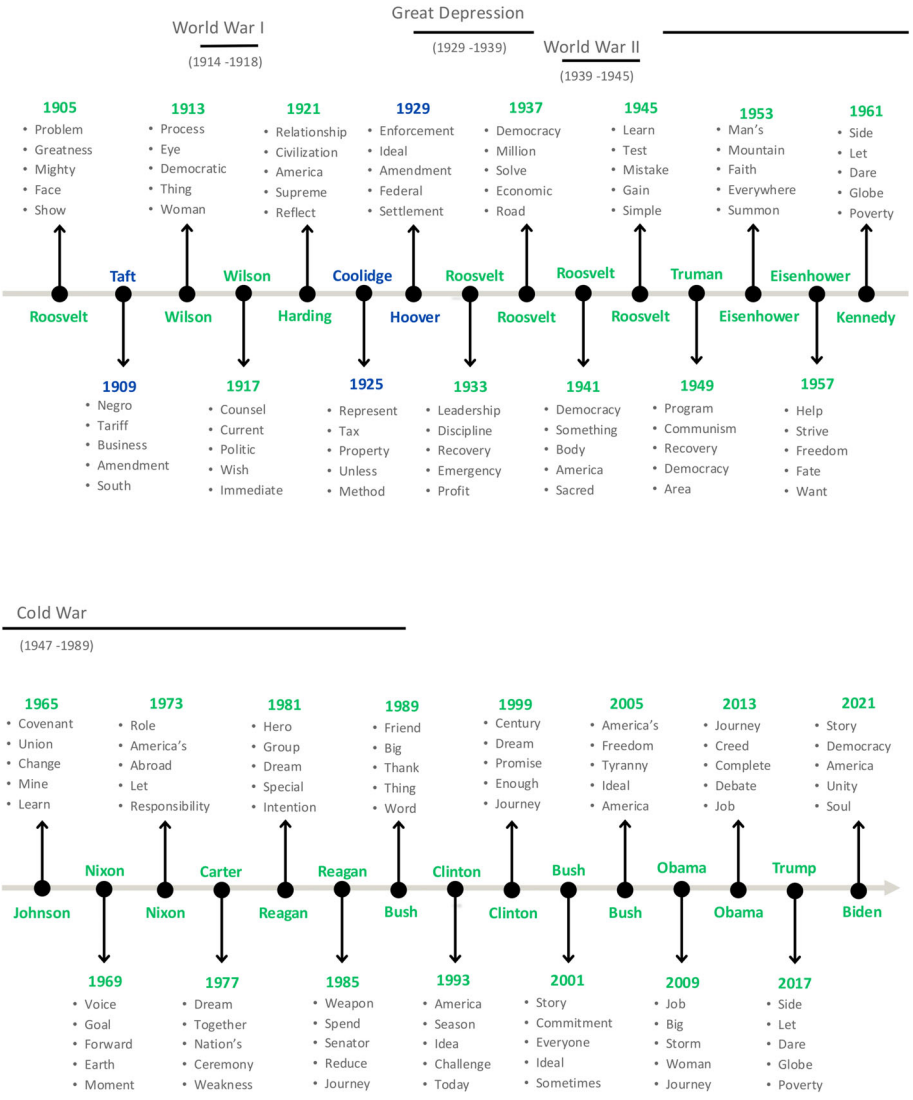


Fig. 4 continued

tices. This implies that these presidents' speeches tend to focus more on the theme of the "American Dream," symbolizing aspirations, progress, and national ideals, while maintaining a connection to political and social issues. An interesting case is Lincoln, whose speeches (such as the one from 1865) occupy positions consistent with a stronger emphasis on unity or policy-related concerns. This places him closer to modern rhetorical changes, acting as a bridge between the two clusters and highlighting his historical and rhetorical significance.

Finally, Fig. 7 examines the 30 most frequent words in each document cluster, colored by word-cluster membership. The second document cluster draws heavily on the "American Dream" word set, whereas the first cluster incorporates terms largely from "Law and Order," as well as some from "Politics, Economics, and Secession." This pattern aligns with the nation's

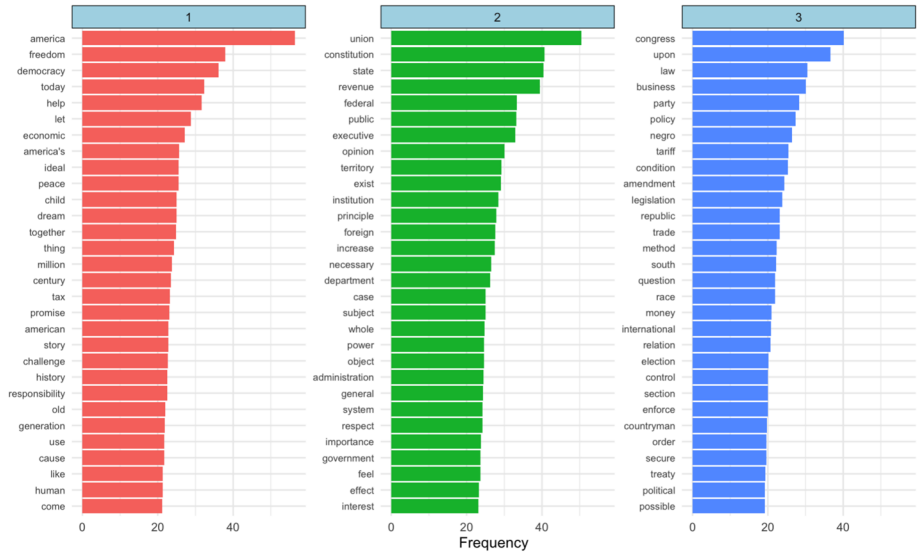


Fig. 5 Top 30 terms in each of the three word clusters (TF-IDF weighted)

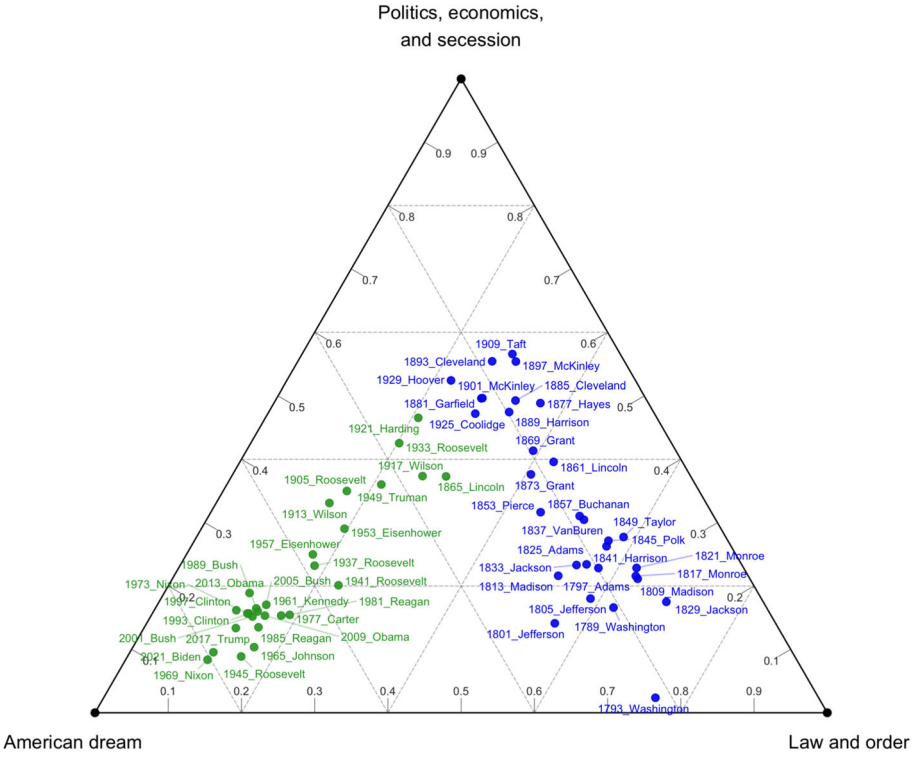


Fig. 6 Each point represents a single inaugural address, colored by its document-cluster membership (blue for cluster 1, green for cluster 2). The coordinates are the mean TF-IDF values of each address in the three word-clusters

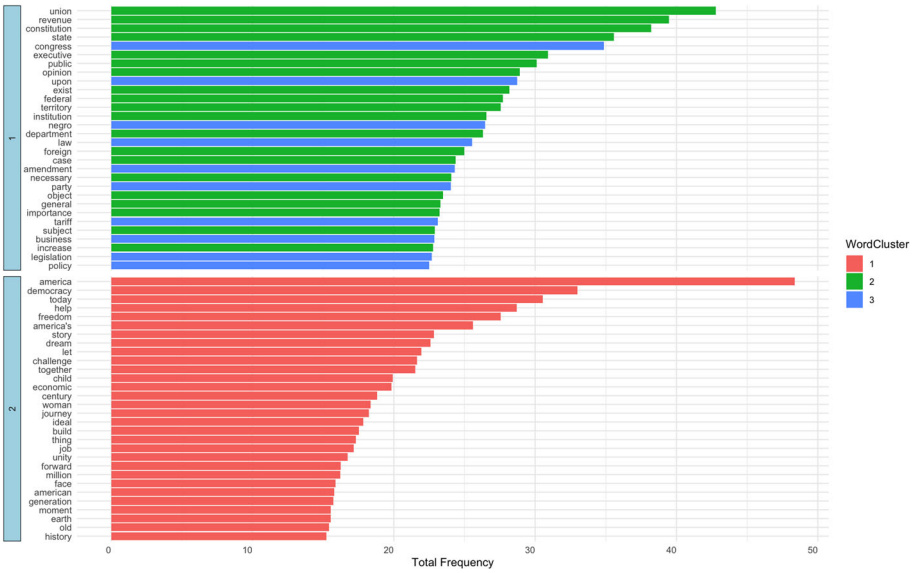


Fig. 7 Most frequent words in each document cluster, colored by their respective word clusters

historical development: early speeches concerned with legal underpinnings and secession disputes, and later addresses emphasizing aspirational, forward-facing themes.

Altogether, these results highlight how the themes covered by the presidents' addresses evolved in time and how major historical events, such as the Civil War, world wars, and the Great Depression, shaped the language and focus of presidential inaugural addresses, illustrating a broader shift from foundational legal concerns to modern themes of national identity and aspirations for the future.

7 Comparison of DKM and SDKM Clusters

We applied double k-means (DKM) on the same data (preprocessed and TF-IDF weighted) to compare the results with those achieved using SDKM. We examine both document-level and word-level clusters, highlighting how Euclidean distance (DKM) versus cosine similarity (SDKM) can produce subtle but meaningful differences in this double clustering setting. We do not dwell on the comparison of computational times, as they are negligible in both methods.

7.1 Document Clusters: Lincoln 1865 as the Only Difference

Regarding document-cluster membership, strikingly, both methods yield the same two-cluster partition (earlier historical presidents vs. later ones), with a single exception: Lincoln's 1865 address. Under DKM, Lincoln's 1865 inaugural is assigned to the older (founding era) cluster because, even with TF–IDF weighting, Euclidean distance remains sensitive to the overall magnitude of the feature vector, making the speech closer to earlier addresses in that sense; by contrast, SDKM assigns Lincoln's 1865 address to the more modern cluster because it

emphasizes relative word-usage patterns (e.g., “union,” “heal,” “mercy,” and “nation”), which makes it thematically closer to later rhetoric.

7.2 Word Clusters: Thematic Comparisons

Beyond their broad similarity in document partitioning, DKM and SDKM exhibit noteworthy differences at the word level. To illustrate how each method groups the main words, we build a contingency table, represented in Table 8, that compares the cluster assignments for all words. Specifically, each row corresponds to a DKM cluster, and each column to an SDKM cluster.

The Table reveals how words are distributed across the six possible cluster pairs. For example, 201 words appear in both cluster 1 under DKM and cluster 1 under SDKM, whereas 256 words assigned to DKM 3 fall into SDKM 1. These differences reflect how each method partitions the same vocabulary, highlighting areas of overlap and divergence. To further investigate differences in the two partitions, we use the ARI to measure the similarity between two partitions; it usually ranges in [0, 1], and it is equal to 1 when the two partitions perfectly match, while it is equal to 0 when one of the two partitions can be considered as a partition obtained by chance. The two partitions are dissimilar, as also the ARI equal to 0.225 reveals.

By exploring differences in the clustering interpretation, we considered the “top words” (words with the 30 highest frequencies) in each cluster. In fact, these words help to label each cluster with a specific topic.

Figure 8 provides a complementary view of these differences. In each panel, we compare the top words identified by DKM and by SDKM for the same cluster index. If a word appears in both methods’ top 30, it is shown in purple (“Overlap”), while words unique to DKM or SDKM are colored differently.

Both DKM and SDKM isolate a cluster of aspirational or national-identity terms, including “America,” “freedom,” “democracy,” “child,” “peace,” and “dream.” These words typically co-occur in modern addresses (post–Civil War and throughout the 20th century), emphasizing a forward-looking vision for the nation. Because such vocabulary dominates many speeches from that era, both Euclidean- and cosine-based approaches identify a similar “American Dream” cluster, which is apparent in Fig. 8 through the substantial overlap (purple bars) in the first panel. The second cluster, highlighted by words like “union,” “constitution,” “state,” and “revenue,” centers on governance and legal structures. These foundational words appear mostly as overlap, reflecting how both methods capture a “Law and Order” theme. Nevertheless, DKM alone includes terms such as “congress” and “law” in its top list, whereas SDKM alone brings in additional administrative or procedural vocabulary such as “case” and “department.” This suggests that SDKM’s emphasis on relative usage encourages a slightly finer partitioning of specialized policy references, whereas DKM merges them when they appear together in large documents. The third cluster, focusing on politics, economic considerations, and older secession-related rhetoric, shows somewhat greater divergence.

Table 8 Contingency table for all words assigned to each pair of clusters (DKM rows, SDKM columns)

DKM\SDKM	1	2	3
1	201	0	0
2	0	295	143
3	256	125	185

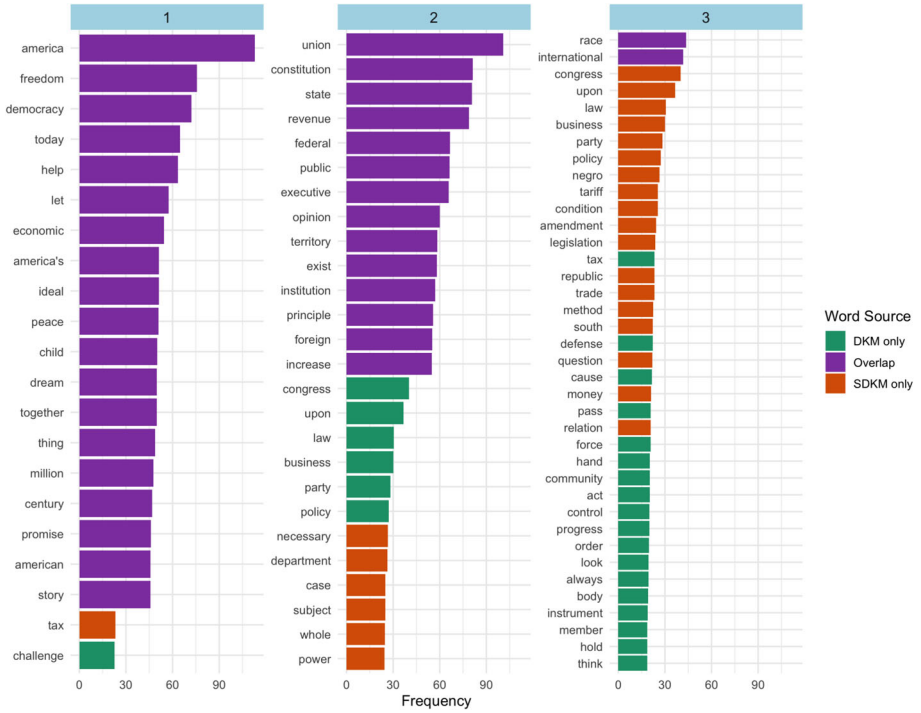


Fig. 8 Top words in each cluster for DKM vs. SDKM. Purple bars indicate overlap in both methods, green indicates DKM only, and orange indicates SDKM only

Figure 8 reveals that SDKM uniquely elevates words like "negro," "tariff," and "amendment," while DKM includes "community," "control," and "progress" in the same top set. Although certain vocabulary, such as "race" and "congress," again appears in both methods' top words, the larger number of unique terms indicates that SDKM splits off certain policy and historical references more decisively, presumably by tracking proportional usage across speeches. DKM, on the other hand, merges them whenever they tend to co-occur in lengthier addresses, thereby placing broad economic and societal concepts together. These patterns underscore how SDKM's cosine-based metric isolates specific rhetorical proportions, creating distinct clusters for governance/constitutional issues versus modern policy/economic themes. Meanwhile, DKM groups some topics together whenever they coexist in the same documents, downplaying subtle proportional distinctions. This aligns with the document-level discrepancy observed in Lincoln's 1865 speech: a borderline case, rich in "union" and "heal" language, can shift clusters under SDKM, just as certain policy or war-related terms shift between the two methods depending on how they recur in the corpus.

8 Comparison with Mixtures of Von Mises–Fisher (vMF) Components

Mixtures of von Mises–Fisher (vMF) distributions provide a natural cosine-based baseline for high-dimensional text, because the vMF density on the unit sphere is $f(x \mid \mu, \kappa) \propto \exp\{\kappa\, \mu^\top x\}$ with $\|\mu\| = 1$ and concentration $\kappa > 0$, so maximum a posteriori hard

assignments are $\arg\max_k \mu_k^\top x$, i.e., they maximize cosine similarity when vectors are ℓ_2-normalized; in other words, vMF is a model-based analog of spherical clustering that uses the same angular geometry as SDKM while not enforcing co-clustering (Banerjee et al., 2005). We therefore fitted vMF to the ℓ_2-normalized term vectors ($K = 3$ components) and to the ℓ_2-normalized document vectors ($Q = 2$ components) using `movMF` R package (Hornik & Grün, 2014), and compared hard assignments to SDKM via the ARI and by inspecting the induced top terms. The document partitions are essentially identical: ARI $= 0.932$ and the contingency table shows a single disagreement (SDKM cluster 1 vs. vMF cluster 1 has 31 speeches, SDKM cluster 2 vs. vMF cluster 2 has 27, with one document swapped), which is consistent with the very large vMF concentrations we estimated at the document level ($\kappa_1 \simeq 879$, $\kappa_2 \simeq 885$) indicating extremely tight spherical components; the top words also match closely across methods for both document clusters, with the "older/governance" group dominated by union, constitution, revenue, state, congress, executive, public, opinion and the "modern/aspirational" group by america, democracy, today, help, freedom, story, dream, together, confirming the SDKM interpretation. For the term partitions the agreement is moderate (ARI $= 0.360$), and the 3×3 cross-tabulation clarifies why: vMF component 2 captures the bulk of SDKM cluster 1 (373 of 457 terms, $\approx 82\%$), whereas the remaining SDKM clusters are split with a "broad" vMF component 1 (276 of 420 from SDKM 2 and 284 of 328 from SDKM 3) plus a "tight" vMF component 3 that carves out a concentrated nucleus from SDKM 2 (131 terms, $\approx 31\%$); this pattern aligns with the estimated vMF concentrations on terms ($\kappa_1 \simeq 50.5$, $\kappa_2 \simeq 45.2$, $\kappa_3 \simeq 104.9$), where component 3 is markedly sharper.

The top-term comparison supports this reading and provides a one-to-one relabelling for interpretation:

1. vMF component 2 aligns with SDKM's "American Dream" word cluster, with overlapping high-loading vocabulary such as time, nation, work, come, long, american, life, know, world, future, hope,
2. vMF component 3 aligns with SDKM's more institutional/policy set (our "Law and Order" axis), concentrating power, may, principle, executive, department, effect, citizen, produce,
3. vMF component 1 aggregates a broader legal/political lexicon that SDKM separates into the "Politics, Economics, and Secession" theme (e.g., law, condition, office, congress, constitution, court, instrument, tariff), reflecting that SDKM's co-clustering leverages the document partition to keep two governance-related vocabularies distinct because they support different groups of speeches, whereas vMF, fitted on rows alone, tends to merge them and then isolate a very coherent administrative core.

The vMF comparison validates SDKM at the document level under a cosine geometry, yielding near-identical partitions and indistinguishable topical summaries, and highlights SDKM's added value for terms: by coupling rows and columns through $\boldsymbol{U}\,\overline{\boldsymbol{Y}}\,\boldsymbol{V}'$, SDKM produces more balanced, block-coherent word clusters that map cleanly onto historical themes, while vMF forms one tight institutional component plus a larger mixed component that blends policy and legal vocabulary. These differences are precisely those expected between a cosine-based co-clustering (SDKM) and a cosine-based model that clusters one mode at a time (vMF).

8.1 Comparison on Simulated Data Scenarios

We further compared SDKM with a cosine-based vMF baseline on the same synthetic scenarios used in Sect. 5, where the true row and column partitions are known ($K_{\text{true}} = 3$, $Q_{\text{true}} = 2$)

Table 9 Mean ARI (rows U, columns V) across $R = 500$ runs by error level ϵ

Method (rows U)	Error level ϵ 0.10	0.35	0.50	0.75	0.90	1.10	1.35	1.50	1.75	2.00
SDKM (mean ARI_U)	1.000	1.000	1.000	0.974	0.907	0.794	0.675	0.587	0.480	0.354
vMF (mean ARI_U)	1.000	1.000	1.000	0.975	0.924	0.825	0.684	0.602	0.477	0.361
Method (columns V)	0.10	0.35	0.50	0.75	0.90	1.10	1.35	1.50	1.75	2.00
SDKM (mean ARI_V)	1.000	1.000	1.000	0.984	0.933	0.840	0.717	0.649	0.507	0.332
vMF (mean ARI_V)	1.000	1.000	1.000	0.977	0.951	0.911	0.850	0.812	0.763	0.692

SDKM values from Table 5; vMF values from the new vMF runs on the same datasets

and the error level ϵ jointly perturbs within–cluster dispersion and centroid heterogeneity. For each $\epsilon \in \{0.10, 0.35, 0.50, 0.75, 0.90, 1.10, 1.35, 1.50, 1.75, 2.00\}$, we generated $R = 500$ datasets. We then fitted SDKM (as in Sect. 5) and mixtures of vMF to ℓ_2–normalized rows and, separately, to the ℓ_2–normalized columns (via `movMF`, 5 random starts), producing hard assignments for rows (U) and columns (V). Performance is reported with the ARI against the true partitions.

Table 9 shows that both methods achieve perfect recovery at low noise. As ϵ increases, row (U) recovery for vMF and SDKM is nearly indistinguishable (vMF slightly higher on average by ≤ 0.03 for $\epsilon \in [0.90, 1.50]$). For columns ($V$), vMF retains higher ARI at moderate/high noise (e.g., mean $\text{ARI}_V = 0.912$ vs. 0.840 at $\epsilon = 1.10$, 0.850 vs. 0.717 at $\epsilon = 1.35$, and 0.692 vs. 0.332 at $\epsilon = 2.00$). Median ARIs tell a similar story: both methods keep median $\text{ARI}_V = 1$ up to $\epsilon \approx 1.35$, after which vMF degrades more gracefully. This pattern is coherent with the data-generating geometry: when clusters are well represented by spherical components per mode, a per-mode vMF mixture is close to a likelihood-correct oracle for partition recovery, while SDKM optimizes a joint co-clustering objective. Complementarily, SDKM estimates a block centroid matrix $\overline{\boldsymbol{Y}}$, for which our simulations showed small NRMSE even as ϵ grows (Table 5), a quantity not directly produced by per-mode vMF. The simulated study confirms that SDKM matches vMF on rows and trades some partition ARI on columns at high noise for the ability to recover coherent biclusters and their block-level centroids.

9 20 Newsgroups Application

We consider the 20 Newsgroups corpus, a benchmark of posts spanning 20 topical Usenet groups (computers, recreation, science, politics, religion). The corpus is freely available online from multiple sources. The posts date to April–May 1993, so the terms reflect the usage and topics of the early-1990s. The document-term matrix of raw messages contains $N = 74,416$ terms and $J = 18,828$ documents. From it we construct a sparse document–term matrix via a preprocessing pipeline aligned with the presidents application: headers and boilerplate are stripped, tokens are lowercased and restricted to alphabetic strings of length 3–14, dictionary lemmatization maps inflected forms to lemmas, and we remove both standard English stopwords and domain–specific metadata terms (e.g., subject, organization, nntp). We then trim features by document frequency (retain terms appearing in at least 1% but in no more than 50% of documents), apply TF–IDF weighting. To keep computation reproducible and the evaluation balanced across topics, we draw a stratified simple random sample of documents within each newsgroup at a rate $p = 0.15$, which preserves the original topic proportions up to rounding. The resulting TF–IDF matrix has $N = 1278$ terms and $J = 2834$ documents and is the input used for SDKM.

To choose (K, Q) we computed the pseudo–F index on a full grid $K, Q \in \{2, \ldots, 22\}$ (included in the supplementary material). The global maximum occurs at $(K, Q) = (2, 2)$. Consistent with our simulation evidence that pseudo–F tends to under–estimate the true cluster counts, we adopt the second-best solution $(K, Q) = (3, 2)$ for interpretability and granularity. With this choice, the final partitions contain 733, 304, and 241 terms across the three term clusters and 1603 and 1231 documents across the two document clusters; the resulting imbalance (one larger term cluster and a mildly dominant document cluster) is typical of sparse text corpora and will be reflected in the interpretation below.

9.1 Cluster Structure and Interpretation

On the **word side**, three distinct vocabularies emerge. **Cluster 1** gathers terms tied to public issues, religion/politics, and some recreation/automotive cues (atheist, muslim, war, bmw, ride, country), consistent with discussions in the social and politics groups and with period cues. **Cluster 2** is a high-frequency, generic discourse layer that cuts across topics (good, can, get, much, know, one), acting as connective vocabulary between more specialized clusters. **Cluster 3** concentrates technical and marketplace lexicon typical of the computing groups (use, sale, drive, mail, window, work), pointing to hardware/software support and classifieds.

On the **document side**, the two clusters align cleanly with topical regimes revealed by smoothed log–odds distinctiveness and by nearest–centroid exemplars. **Document Cluster 1** aggregates religion/politics and sports chatter (Jew, Jesus, Waco, Israel, playoff, league), with exemplars from `soc.religion.christian`, `talk.politics.`, `talk.religion.misc`, and `rec.sport.`. **Document Cluster 2** is dominated by computing and for-sale/technical help (scsi, window, ide, screen, port, cpu); its most representative posts come from `comp_*`, `misc.forsale`, and `sci.electronics`.

In the co-clustering view, Word Cluster 3 loads most on Document Cluster 2 (computing/marketplace), Word Cluster 1 on Document Cluster 1 (religion, politics, sports), while Word Cluster 2 provides cross-cutting discourse shared by both document clusters.

10 Final Considerations

In this paper, we introduced the spherical double k-means clustering method, a novel approach for the simultaneous partitioning of terms and documents in textual data. By integrating the principles of double k-means, double k-means, and spherical k-means clustering, SDKM effectively addresses the challenges posed by high dimensionality, sparsity, and noise inherent in textual data. The incorporation of cosine similarity within the co-clustering framework allows for a deep understanding of the text and a more meaningful clustering by considering the angular relationships between data points, which is particularly beneficial for textual data where document lengths and term frequencies can vary significantly. In fact, traditional k-means using Euclidean distance often produces large, dominant clusters alongside smaller residual ones when applied to textual data, even after TF-IDF weighting. This tendency occurs because Euclidean distance is most effective when variables are uncorrelated, a condition seldom met in language data, where terms frequently co-occur in highly correlated patterns. By contrast, SDKM, employing cosine similarity, focuses on the relative distribution of term usage rather than absolute frequencies. In doing so, it captures the angular relationships between document vectors, thereby disentangling topics more effectively and yielding clusters that are more semantically coherent. Applying SDKM to the US presidential inaugural addresses dataset demonstrated its capability to uncover distinct thematic clusters that evolved over time. The method not only identified clear chronological distinctions in the clusters of documents but also revealed how historical events have influenced the themes and language used in presidential discourse. For instance, the transition between clusters aligns with significant periods such as the civil war, the great depression, and world wars, highlighting shifts in national priorities and rhetoric. This application underscores the potential of SDKM to extract meaningful patterns from complex text corpora, making it a valuable tool for tasks such as topic modeling, sentiment analysis, and information retrieval.

In addition to demonstrating SDKM's effectiveness on the presidential addresses corpus, we compared its results with double k-means using the same TF-IDF weighted data. While both methods identified a broadly consistent chronological split for most documents, one notable discrepancy emerged with Lincoln's 1865 speech, which SDKM grouped among more modern addresses. At the word level, the methods agreed on an "American Dream" cluster but diverged significantly on governance and policy-related terms, indicating that SDKM's cosine-based co-clustering can isolate finer proportional distinctions. These findings reinforce SDKM's potential to uncover nuanced patterns in complex textual corpora, complementing more traditional Euclidean-based approaches.

Despite the effectiveness of SDKM, there are avenues for further improvement and research. One area of future development is the implementation of a fuzzy version of SDKM. The current method assigns each term and document exclusively to one cluster, which may not fully capture the nuanced relationships in textual data where terms and documents can naturally belong to multiple topics or themes. A fuzzy clustering approach would allow for degrees of membership, providing a more flexible and realistic representation of the data's inherent structure. For future work in this direction, we may consider a finite-mixture approach, for example, the one implemented in the `R` package `movMF` (Hornik & Grün, 2014).

An open issue relates to the normalization of the centroid matrix within the co-clustering framework. Ideally, to achieve spherical clusters in both dimensions, the centroid matrix should be normalized by both rows and columns. However, simultaneous normalization poses challenges due to the interdependence of term and document clusters in co-clustering. In our approach, we opted to normalize the centroid matrix by rows, ensuring that the term vectors (rows) have unit length. This decision was based on the consideration that terms typically exhibit more variability and noise than documents. By focusing on the sphericality of term clusters, we enhance the method's robustness against noise in the data. While the norms of the columns (documents) in the centroid matrix were not explicitly normalized, they were observed to be close to one, suggesting that the lack of column normalization may not significantly impact the clustering results. Future work could explore alternative normalization strategies or iterative normalization procedures to address this issue more comprehensively.

Appendix 1. Monotonicity Proof for SDKM

This appendix demonstrates that the SDKM objective function does not decrease at each iteration, ensuring convergence to a local maximum. Specifically, if $\left(\boldsymbol{U}^{(t)}, \overline{\boldsymbol{Y}}^{(t)}, \boldsymbol{V}^{(t)}\right)$ denotes the solution at iteration t, we want to show

$$f\left(\boldsymbol{U}^{(t)}, \overline{\boldsymbol{Y}}^{(t)}, \boldsymbol{V}^{(t)}\right) \;\leq\; f\left(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+1)}, \boldsymbol{V}^{(t+1)}\right),$$

where

$$f(\boldsymbol{U}, \overline{\boldsymbol{Y}}, \boldsymbol{V}) \;=\; \mathrm{tr}\left(\boldsymbol{X}' \boldsymbol{U}\, \overline{\boldsymbol{Y}}\, \boldsymbol{V}'\right).$$

By proving that each sub-step (updating $\boldsymbol{U}$, then $\overline{\boldsymbol{Y}}$, then $\boldsymbol{V}$, then $\overline{\boldsymbol{Y}}$ again) does not reduce f, we establish that the full iteration is monotonically non-decreasing.

With $\pi_k^{(t)}$ is denoted the set of row observations (units) assigned to cluster k under $\boldsymbol{U}^{(t)}$. $\tau_q^{(t)}$ denotes the set of column features assigned to cluster q under $\boldsymbol{V}^{(t)}$. Each row $\overline{\boldsymbol{x}}_k$ in $\overline{\boldsymbol{Y}}$ is viewed as the row-centroid for cluster k.

At iteration t, SDKM updates:

$$(\boldsymbol{U}^{(t)}, \overline{\boldsymbol{Y}}^{(t)}, \boldsymbol{V}^{(t)}) \;\rightarrow\; (\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+1)}, \boldsymbol{V}^{(t+1)}),$$

by successively maximizing the objective with respect to $\boldsymbol{U}$, then $\overline{\boldsymbol{Y}}$, then $\boldsymbol{V}$, and finally $\overline{\boldsymbol{Y}}$ again.

Step 1: Updating $\boldsymbol{U}$.

$$f\big(\boldsymbol{U}^{(t)}, \overline{\boldsymbol{Y}}^{(t)}, \boldsymbol{V}^{(t)}\big) \;=\; \mathrm{tr}\big(\boldsymbol{X}'\,\boldsymbol{U}^{(t)}\,\overline{\boldsymbol{Y}}^{(t)}\,\boldsymbol{V}^{(t)}\big).$$

Because $\boldsymbol{U}^{(t+1)}$ is chosen to maximize the objective for fixed $\overline{\boldsymbol{Y}}^{(t)}$ and $\boldsymbol{V}^{(t)}$, we have

$$f\big(\boldsymbol{U}^{(t)}, \overline{\boldsymbol{Y}}^{(t)}, \boldsymbol{V}^{(t)}\big) \;\leq\; f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t)}, \boldsymbol{V}^{(t)}\big). \tag{18}$$

Step 2: Updating $\overline{\boldsymbol{Y}}$.

Holding $\boldsymbol{U}^{(t+1)}$ and $\boldsymbol{V}^{(t)}$ fixed, we update $\overline{\boldsymbol{Y}}$ to $\overline{\boldsymbol{Y}}^{(t+1)}$ by maximizing $\mathrm{tr}(\boldsymbol{X}'\,\boldsymbol{U}^{(t+1)}\,\overline{\boldsymbol{Y}}\,\boldsymbol{V}^{(t)})$. Consequently,

$$f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t)}, \boldsymbol{V}^{(t)}\big) \;\leq\; f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+1)}, \boldsymbol{V}^{(t)}\big). \tag{19}$$

Combining Eqs. 18 and 19 yields

$$f\big(\boldsymbol{U}^{(t)}, \overline{\boldsymbol{Y}}^{(t)}, \boldsymbol{V}^{(t)}\big) \;\leq\; f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+1)}, \boldsymbol{V}^{(t)}\big).$$

Step 3: Updating $\boldsymbol{V}$.

Now, for fixed $\boldsymbol{U}^{(t+1)}$ and $\overline{\boldsymbol{Y}}^{(t+1)}$, updating $\boldsymbol{V}^{(t)} \rightarrow \boldsymbol{V}^{(t+1)}$ once again does not lower the objective:

$$f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+1)}, \boldsymbol{V}^{(t)}\big) \;\leq\; f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+1)}, \boldsymbol{V}^{(t+1)}\big). \tag{20}$$

Step 4: Final re-update of $\overline{\boldsymbol{Y}}$.

After $\boldsymbol{V}$ is updated, the algorithm updates the centroids $\overline{\boldsymbol{Y}}^{(t+1)}$ to $\overline{\boldsymbol{Y}}^{(t+2)}$. Again, by maximizing $\mathrm{tr}(\boldsymbol{X}'\,\boldsymbol{U}^{(t+1)}\,\overline{\boldsymbol{Y}}\,\boldsymbol{V}^{(t+1)})$, we have

$$f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+1)}, \boldsymbol{V}^{(t+1)}\big) \;\leq\; f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+2)}, \boldsymbol{V}^{(t+1)}\big). \tag{21}$$

Chain of inequalities and conclusion.

Combining all these inequalities, Eqs. 18, 19, 20, and 21, we obtain:

$$\begin{aligned} f\big(\boldsymbol{U}^{(t)}, \overline{\boldsymbol{Y}}^{(t)}, \boldsymbol{V}^{(t)}\big) \;&\leq\; f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+1)}, \boldsymbol{V}^{(t)}\big) \;\leq\; f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+1)}, \boldsymbol{V}^{(t+1)}\big) \\ &\leq\; f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+2)}, \boldsymbol{V}^{(t+1)}\big). \end{aligned}$$

Hence,

$$f\big(\boldsymbol{U}^{(t+1)}, \overline{\boldsymbol{Y}}^{(t+2)}, \boldsymbol{V}^{(t+1)}\big) \;\geq\; f\big(\boldsymbol{U}^{(t)}, \overline{\boldsymbol{Y}}^{(t)}, \boldsymbol{V}^{(t)}\big).$$

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00357-026-09544-7.

Acknowledgements The views and opinions expressed are those of the authors and do not necessarily reflect the official policy or position of the Italian National Institute of Statistics - Istat.

Funding Open access funding provided by Università degli Studi di Roma Tor Vergata within the CRUI-CARE Agreement.

Data and Code Availability All analyses presented in this article are reproducible using the data and code publicly available at https://github.com/eseri93/SDKM.

Declarations

Competing Interests The authors declare no competing interests.

References

Aggarwal, C.C., & Zhai, C. (2012). *Mining text data*. Springer. US. https://doi.org/10.1007/978-1-4614-3223-4

Aggarwal, C.C., Hinneburg, A., & Keim, D.A. (2001). On the surprising behavior of distance metrics in high dimensional space. In: *International conference on database theory*, Springer, (pp. 420–434).

Banerjee, A., Dhillon, I. S., Ghosh, J., et al. (2005). Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, *6*(46), 1345–1382. http://jmlr.org/papers/v6/banerjee05a.html.

Benoit, K., Watanabe, K., Wang, H., et al. (2018). quanteda: An r package for the quantitative analysis of textual data. *Journal of Open Source Software*, *3*(30), 774. https://doi.org/10.21105/joss.00774, https://quanteda.io

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research, 3*(Jan), 993–1022. https://doi.org/10.5555/944919.944937

Bouveyron, C., Celeux, G., Murphy, T. B., et al. (2019). *Model-based clustering and classification for data science: With applications in R*. Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, Cambridge, England. https://doi.org/10.1017/9781108644181

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics, 3*(1), 1–27. https://doi.org/10.1080/03610927408827101

Celardo, L., Iezzi, D.F., & Vichi, M. (2016). Multi-mode partitioning for text clustering to reduce dimensionality and noises. In: JADT 2016-statistical analysis of textual data. Presses de FacImprimeur, (vol. 1, pp. 181–192).

Churchill, R., & Singh, L. (2022). The evolution of topic modeling. *ACM Computing Surveys*, *54*(10s). https://doi.org/10.1145/3507900.

Cozzolino, I., & Ferraro, M. B. (2022). Document clustering. *Wiley Interdisciplinary Reviews: Computational Statistics, 14*(6), e1588. https://doi.org/10.1002/wics.1588

Dhillon, I.S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, (pp. 269–274).

Dhillon, I. S., & Modha, D. S. (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning, 42*, 143–175. https://doi.org/10.1023/A:1007612920971

Di Nuzzo, C., & Ingrassia, S. (2022). A mixture model approach to spectral clustering and application to textual data. *Statistical Methods & Applications, 31*(5), 1071–1097.

Duwairi, R., & Abu-Rahmeh, M. (2015). A novel approach for initializing the spherical k-means clustering algorithm. *Simulation Modelling Practice and Theory, 54*, 49–63. https://doi.org/10.1016/j.simpat.2015.03.007

Fraley, C., & Raftery, A. E. (1998). How many clusters? Which clustering method? Answers via model-based cluster analysis. *The Computer Journal, 41*(8), 578–588. https://doi.org/10.1093/comjnl/41.8.578

Hornik, K., Feinerer, I., Kober, M., et al. (2012). Spherical k-means clustering. *Journal of Statistical Software, 50*, 1–22. https://doi.org/10.18637/jss.v050.i10

Hornik, K., & Grün, B. (2014). movMF: An r package for fitting mixtures of von mises-fisher distributions. *Journal of Statistical Software, 58*, 1–31.

Huang, A., et al. (2008). Similarity measures for text document clustering. In: *Proceedings of the sixth New Zealand computer science research student conference (NZCSRSC2008)*, Christchurch, New Zealand, (pp. 9–56).

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification, 2*, 193–218.

Iezzi DF, Celardo L (2020) Text analytics: Present, past and future. In: D. F. Iezzi, D. Mayaffre, M. Misuraca, (eds.), *Text Analytics. Springer International Publishing*, Cham, (pp. 3–15). https://doi.org/10.1007/978-3-030-52680-1_1

Iezzi, D. F. (2012). Centrality measures for text clustering. *Communications in Statistics - Theory and Methods, 41*(16–17), 3179–3197. https://doi.org/10.1080/03610926.2011.633729

Janani, R., & Vijayarani, S. (2019). Text document clustering using spectral clustering algorithm with particle swarm optimization. *Expert Systems with Applications, 134*, 192–200.

Kim, H., Kim, H. K., & Cho, S. (2020). Improving spherical k-means for document clustering: Fast initialization, sparse centroid projection, and efficient cluster labeling. *Expert Systems with Applications, 150*(113), 288. https://doi.org/10.1016/j.eswa.2020.113288

Kluger, Y., Basri, R., Chang, J. T., et al. (2003). Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Research, 13*(4), 703–716.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature, 401*(6755), 788–791.

Liang, M., & Niu, T. (2022). Research on text classification techniques based on improved TF-IDF algorithm and LSTM inputs. *Procedia Computer Science*, *208*, 460–470. https://doi.org/10.1016/j.procs.2022.10.064, https://www.sciencedirect.com/science/article/pii/S1877050922015058. 7th International Conference on Intelligent, Interactive Systems and Applications

Lodhi, H., Saunders, C., Shawe-Taylor, J., et al. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, *2*(Feb), 419–444. https://doi.org/10.1162/153244302760200687

Maher, K., & Joshi, M. S. (2016). Effectiveness of different similarity measures for text classification and clustering. *International Journal of Computer Science and Information Technologies, 7*(4), 1715–1720.

Manning, C. D. (2009). *An introduction to information retrieval*. Cambridge University Press.

McQueen, J.B. (1967). Some methods of classification and analysis of multivariate observations. In: *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, (pp. 281–297).

Mehta, V., Agarwal, M., & Kaliyar, R. K. (2024). A comprehensive and analytical review of text clustering techniques. *International Journal of Data Science and Analytics, 18*(3), 239–258. https://doi.org/10.1007/s41060-024-00540-x

Rocci, R., & Vichi, M. (2008). Two-mode multi-partitioning. *Computational Statistics & Data Analysis, 52*(4), 1984–2003. https://doi.org/10.1016/j.csda.2007.06.025

Steinbach, M., Karypis, G., & Kumar, V. (2000). *A comparison of document clustering techniques*. Tech. rep., University of Minnesota. https://www.stat.cmu.edu/~rnugent/PCMI2016/papers/DocClusterComparison.pdf.

Vichi, M. (2015). Two-mode partitioning and multipartitioning. In: *Handbook of Cluster Analysis*. Chapman and Hall/CRC, (pp. 540–565).

Vichi, M. (2001). Double k-means clustering for simultaneous classification of objects and variables. In S. Borra, R. Rocci, & M. Vichi (Eds.), *Advances in Classification and Data Analysis* (pp. 43–52). Berlin Heidelberg, Berlin, Heidelberg: Springer.

Zhao, Y., & Karypis, G. (2004). Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning, 55*, 311–331. https://doi.org/10.1023/B:MACH.0000027785.44527.d6

Zhong, S., & Ghosh, J. (2005). Generative model-based document clustering: A comparative study. *Knowledge and Information Systems, 8*(3), 374–384.