Topic Modeling: BERTopic

Lorenzo Mannocci

a.a. 2026/2027



UNIVERSITÀ DI PISA

Discovering topics in text collections

Text collections contain information that is difficult to explore manually:

- News articles describe events and subjects.
- Customer reviews discuss experiences and product features.
- Scientific abstracts present research problems and methods.

Goal: automatically discover recurring themes in a collection of documents.

Topic modeling

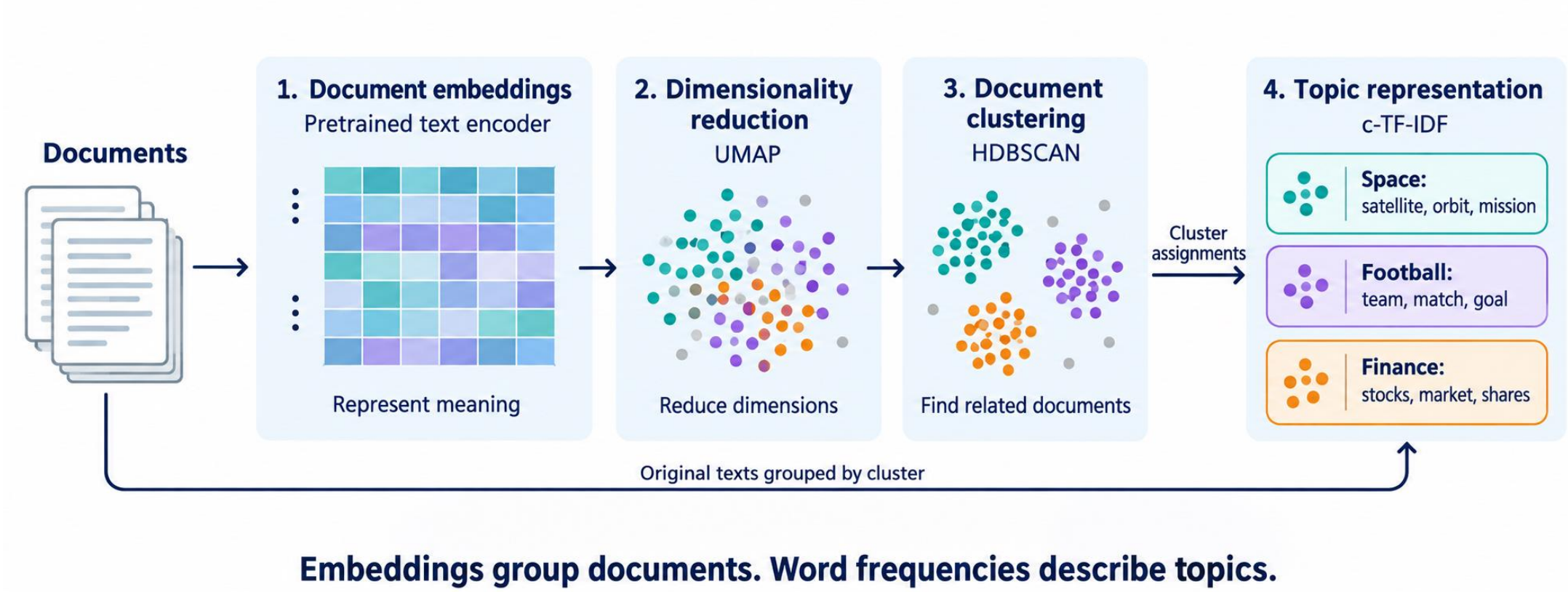
Topic modeling aims to discover recurring themes in a collection of documents.

A topic can be described by a set of representative words and illustrated through relevant documents.

In unsupervised topic discovery, the topic categories are not provided in advance.

Example documents	**Representative words**	**Interpreted theme**
"A satellite entered orbit." / "The spacecraft began its mission."	satellite, orbit, spacecraft, mission	Space exploration
"The team won the match." / "The striker scored two goals."	team, match, striker, goals	Football
"Share prices increased." / "Investors reacted to market uncertainty."	shares, prices, investors, market	Financial markets

BERTopic pipeline



1. Convert each document into a numerical vector capturing semantic information.
2. Extract representative words from the original texts in each cluster.
3. Construct a smaller representation for clustering.
4. Identify groups of related documents and mark outliers.

Grootendorst, M. (2022). **BERTopic: Neural topic modeling with a class-based TF-IDF procedure**. *arXiv preprint arXiv:2203.05794*.

Embeddings

Representing text with word counts

Clustering algorithms require numerical representations.

Bag-of-words: represent each document through the frequencies of its words.

Document	satellite	entered	orbit	team	won	match
"The satellite entered orbit."	1	1	1	0	0	0
"The team won the match."	0	0	0	1	1	1
"The satellite."	1	0	0	0	0	0

Example after lowercasing, removing punctuation, and excluding "the".

For n documents and a vocabulary of V terms:

$$X \in \mathbb{R}^{n \times V}.$$

- Each row represents a document;
- Each column represents a vocabulary term;
- Most entries are zero: the representation is sparse;
- Discards word order: sentences containing the same words have the same representation, even when their meanings differ.

Word overlap and semantic similarity

Consider two sentences:

- "The car needs repairs."
- "The automobile requires maintenance."

They express similar meanings but share few words.

Limitations of word-count representations

- Different words can express similar meanings: car / automobile.
- The same word can have different meanings: river bank / investment bank.
- Word order and context affect meaning.
- TF-IDF reweights words according to their frequency in a document and their prevalence across documents. It still relies on vocabulary overlap.

Document embeddings aim to represent semantic relationships beyond exact word matching.

From documents to embeddings

A document embedding is a dense numerical vector representing information about the meaning of a text.

A pretrained text encoder maps each document d_i to a vector:

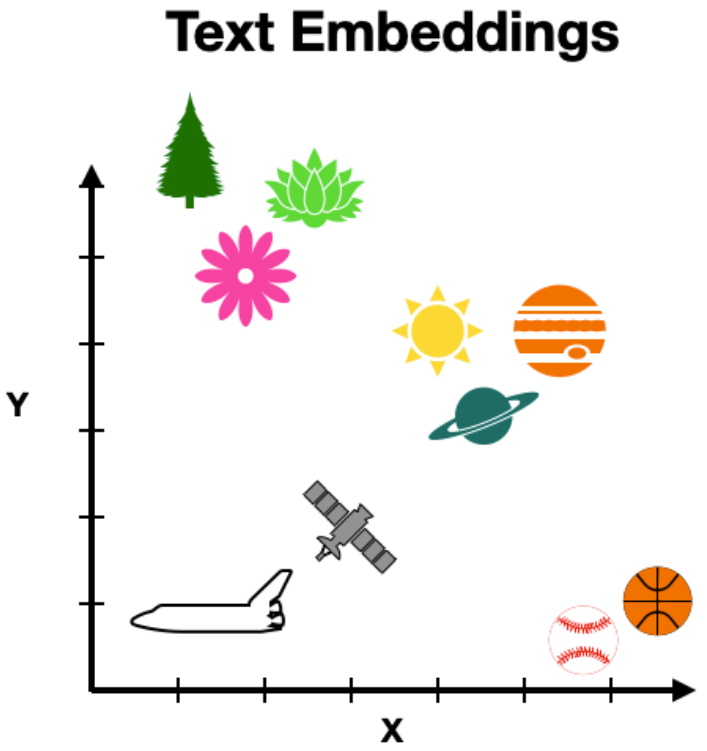
$$x_i = f_\theta(d_i) \in \mathbb{R}^p$$

- Each document becomes a vector of the same dimension p.
- The encoder uses the words and their context.
- Semantically related texts are expected to have similar embeddings.

For ϕ documents:

$$X \in \mathbb{R}^{n \times p}$$

Unlike word-count features, individual embedding dimensions generally have no simple human-readable meaning.



BERT: representing words in context

BERT (Bidirectional Encoder Representations from Transformers) is a pretrained language model that produces contextual representations of text.

It processes a sequence of tokens, which can be words or parts of words.
Each token's representation incorporates information from the surrounding text.
Its bidirectional encoder uses context from both before and after a token.

The same word can therefore receive different representations in different contexts.

Text	**Interpretation of "bank"**
"They sat on the river bank."	Land beside a river
"They deposited money in the bank."	Financial institution

BERT produces a vector for each input token.
To represent an entire document, these vectors must be combined into a single vector.

Grootendorst, M. (2022). **BERTopic: Neural topic modeling with a class-based TF-IDF procedure**. *arXiv preprint arXiv:2203.05794*.

From BERT to Sentence Transformers

For clustering, we need one vector per document, with distances that reflect useful semantic relationships.

Sentence-BERT (SBERT) adapts a pretrained transformer encoder to produce embeddings suitable for sentence similarity and clustering. It trains on relationships between texts so that related texts receive similar vectors.

The encoding process consists of:

1. Transformer encoder: Contextual vectors for the input tokens;
2. Pooling: One vector summarizing the text. A common pooling strategy averages the contextual token vectors, excluding padding positions.

Sentence Transformers is a library providing pretrained embedding models based on different transformer architectures.

BERTopic commonly uses these models. For example, all-MiniLM-L6-v2 maps sentences and paragraphs to 384-dimensional vectors. Despite its name, BERTopic does not require the original BERT model.

Grootendorst, M. (2022). **BERTopic: Neural topic modeling with a class-based TF-IDF procedure**. *arXiv preprint arXiv:2203.05794*.
Reimers, N., & Gurevych, I. (2019, November). **Sentence-bert: Sentence embeddings using siamese bert-networks**. In *EMNLP-IJCNLP* (pp. 3982-3992).

Output of embeddings

Once documents are represented as vectors, we can compare them numerically. Encoding n documents produces a numerical matrix:

$$X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix} \in \mathbb{R}^{n \times p}$$

In the standard BERTopic workflow, the encoder is already trained: we compute embeddings for our corpus without retraining it.

The resulting vectors become the input to dimensionality reduction.

Their quality depends on how well the encoder represents the corpus's language and subject matter.

Grootendorst, M. (2022). **BERTopic: Neural topic modeling with a class-based TF-IDF procedure**. *arXiv preprint arXiv:2203.05794*.

UMAP: dimensionality reduction

From PCA to UMAP

UMAP (Uniform Manifold Approximation and Projection) is a nonlinear dimensionality reduction method.

Before clustering, BERTopic usually constructs a lower-dimensional representation:

$$X \in \mathbb{R}^{n \times p} \longrightarrow Z \in \mathbb{R}^{n \times q}, \quad q \ll p$$

- Distance and density estimation can become less informative in high dimensions.
- Redundant or irrelevant variation can obscure useful structure.
- A smaller representation can reduce the computational cost of clustering.

PCA: it finds a linear projection that preserves as much variance as possible.

UMAP: documents with similar embeddings should remain neighbors after reduction.

PCA	**UMAP**
Finds a linear subspace preserving maximum variance	Constructs an embedding that approximately preserves neighborhood relationships
Uses principal directions	Uses a weighted neighborhood graph
Suited to linear structure	Can represent nonlinear structure

McInnes, L., Healy, J., & Melville, J. (2018). **UMAP: Uniform manifold approximation and projection for dimension reduction**. *arXiv preprint arXiv:1802.03426*.

UMAP

1. Identify neighbors: Find the nearest neighbors of each document in the original embedding space.

2. Build a weighted graph: Represent documents as nodes and neighborhood relationships as weighted edges.

3. Optimize a lower-dimensional representation: Arrange the points so that their relationships approximately match the graph.

Strong connections encourage points to stay close; repulsion prevents the representation from collapsing.

McInnes, L., Healy, J., & Melville, J. (2018). **UMAP: Uniform manifold approximation and projection for dimension reduction**. *arXiv preprint arXiv:1802.03426*.

Building the neighborhood graph

For each point $\mathbf{x}_i$, identify a neighborhood $N_k(i)$ containing its nearest neighbors.

Distances depend on the input representation. For document embeddings, a common choice is cosine distance:

$$d(x_i, x_j) = 1 - \frac{x_i^\top x_j}{\|x_i\|\|x_j\|}$$

Document pair	Similarity
"A satellite entered orbit." "A spacecraft is orbiting Earth."	Higher
"A satellite entered orbit." "The team won the match."	Lower

The neighborhood graph contains:

- Nodes: documents represented by their embeddings.
- Edges: connections between neighboring documents.
- Weights: strengths of these local connections.

The neighborhood size determines the scale of structure considered:

- Smaller neighborhoods emphasize local relationships.
- Larger neighborhoods incorporate relationships over broader regions.

Distances must be interpreted locally: a distance that indicates a close neighbor in a sparse region may indicate a distant neighbor in a dense region.

UMAP therefore adapts the distance scale separately around each point.

McInnes, L., Healy, J., & Melville, J. (2018). **UMAP: Uniform manifold approximation and projection for dimension reduction**. *arXiv preprint arXiv:1802.03426*.

Assigning connection strengths

Let $N_k(i)$ be the neighborhood of i, UMAP computes a directed connection strength:

$$w_{j|i} = \begin{cases} \exp\left(-\dfrac{\max\{0, d(x_i, x_j) - \rho_i\}}{\sigma_i}\right), & j \in N_k(i) \\ 0, & j \notin N_k(i) \end{cases}$$

- ρ_i: **local distance offset**, typically the distance to the nearest distinct neighbor under the standard setting. Subtracting it makes the closest neighbor a reference point with adjusted distance zero;
- σ_i**: local scale fitted** from the neighborhood distances. This controls how quickly the strength decreases:
  - A small σ_i produces faster decay.
  - A large σ_i produces slower decay.
- $w_{j|i}$: **strength of the connection** from point i to point j

The negative exponential converts the adjusted distance into a smoothly decreasing connection strength.

If the distance is at most ρ_i, the result is zero. The connection strength is therefore: $w_{j|i} = e^0 = 1$

McInnes, L., Healy, J., & Melville, J. (2018). **UMAP: Uniform manifold approximation and projection for dimension reduction**. *arXiv preprint arXiv:1802.03426*.

Assigning connection strengths

Nearby points receive strong connections; connection strength decays as distance increases beyond ρ_i.

The local scales make these strengths asymmetric, because ρ_i and σ_i depend on the reference point, so generally $w_{j|i} \neq w_{i|j}$.

UMAP combines both directions using:

$$w_{ij} = w_{j|i} + w_{i|j} - w_{j|i}w_{i|j}$$

This produces a symmetric weighted graph, with $w_{ij} \in [0,1]$.

McInnes, L., Healy, J., & Melville, J. (2018). **UMAP: Uniform manifold approximation and projection for dimension reduction**. *arXiv preprint arXiv:1802.03426*.

Representing connections in fewer dimensions

UMAP uses stochastic gradient updates to adjust the low-dimensional coordinates $\mathbf{z}_i \in \mathbb{R}^q$, making the low-dimensional connection strengths v_{ij} reflect the original strengths w_{ij}.

In the usual Euclidean output space, the connection strength between two embedded points is modeled as:

$$v_{ij} = \frac{1}{1 + a\|z_i - z_j\|^{2b}}, \quad a, b > 0$$

- Nearby embedded points have v_{ij} close to 1.
- Distant embedded points have v_{ij} close to 0.
- Constants a and b control how strength decreases with distance.

These constants are fitted according to the desired spacing behavior of the embedding.

The task is now to find coordinates whose connection strengths v_{ij} agree with the original graph weights w_{ij}.

The optimization adjusts the coordinates; the original graph weights remain fixed.

McInnes, L., Healy, J., & Melville, J. (2018). **UMAP: Uniform manifold approximation and projection for dimension reduction**. *arXiv preprint arXiv:1802.03426*.

Optimizing the embedding

The theoretical graph-matching objective can be expressed, omitting terms constant with respect to the coordinates, as a binary cross-entropy:

$$L = -\sum_{i<j} \left[w_{ij} \log v_{ij} + \left(1 - w_{ij}\right) \log\left(1 - v_{ij}\right) \right]$$

Encourages strongly connected points to remain close

Discourages weakly connected points from collapsing together

McInnes, L., Healy, J., & Melville, J. (2018). **UMAP: Uniform manifold approximation and projection for dimension reduction**. *arXiv preprint arXiv:1802.03426*.

Optimizing the embedding

UMAP initializes the coordinates, commonly using a spectral embedding of the graph, and iteratively updates them.

For efficiency, the implementation uses stochastic updates and negative sampling, rather than explicitly evaluating every pair in the full objective. This is a sampled optimization procedure, not an exact computation of the all-pairs loss at every iteration.

It minimizes a cross-entropy objective, balancing:

- Attraction: strongly connected points in the original space should remain close.
- Repulsion: weakly connected or unconnected points should not all collapse together.
- Approximating the original neighborhood structure.

McInnes, L., Healy, J., & Melville, J. (2018). **UMAP: Uniform manifold approximation and projection for dimension reduction**. *arXiv preprint arXiv:1802.03426*.

Limits of UMAP

UMAP aims to retain useful neighborhood relationships, but dimensionality reduction can distort the data:

- Exact distances are not preserved.
- Relative densities can change because of local distance scaling and embedding optimization.
- Apparent gaps may be amplified, so separated regions do not automatically establish distinct topics.
- The output dimensions have no direct semantic interpretation.
- Because HDBSCAN relies on density, changes introduced by UMAP can affect the clusters it discovers.

In BERTopic, clustering operates on the reduced vectors:

$$x_i \in \mathbb{R}^p \longmapsto z_i \in \mathbb{R}^q \longmapsto \text{cluster assignment}$$

McInnes, L., Healy, J., & Melville, J. (2018). **UMAP: Uniform manifold approximation and projection for dimension reduction**. *arXiv preprint arXiv:1802.03426*.

HDBSCAN: clustering

HDBSCAN: clustering across density levels

DBSCAN uses a single neighborhood radius ε. When clusters have different densities, one radius may not recover all of them satisfactorily:

- A small radius can identify a dense group but fragment a sparse group.
- A large radius can recover the sparse group but merge nearby groups.

HDBSCAN builds a hierarchy across distance thresholds and selects groups that persist.

It combines three ideas:

- Local density: distinguish dense regions from sparse connections;
- Hierarchical clustering: track how groups split as the distance threshold decreases;
- Cluster stability: select persistent groups from the hierarchy.

The number of clusters does not need to be specified beforehand, and some points can remain noise.

Campello, R. J., Moulavi, D., & Sander, J. (2013). **Density-based clustering based on hierarchical density estimates**. In *PAKDD* (pp. 160-172).

Core distance

We characterize density around a point, by computing how far must we go to reach k other points. This distance is the core distance:

$$\text{core}_k(x) = d\big(x, k\text{NN}(x)\big)$$

Θ: the min_samples parameter, which controls how many neighbors we require to consider a point a core point.

Here, $kNN(x)$ denotes the $k-th$ nearest neighbor, excluding x itself.

- Small core distance: many points are nearby.
- Large core distance: the neighborhood is sparse.

The choice of k determines the scale at which local density is assessed.

Example with $k = 3$:

Point	Distances to its three nearest neighbors	Core distance
A	$0.1,\ 0.2,\ 0.3$	0.3
B	$0.4,\ 0.8,\ 1.2$	1.2

Point B requires a larger neighbaorhood to include the same number of neighbors.

Campello, R. J., Moulavi, D., & Sander, J. (2013). **Density-based clustering based on hierarchical density estimates**. In *PAKDD* (pp. 160-172).

Mutual reachability: density-aware distance metric

Two points may be close even when one lies in a sparse region. Such points can create bridges between otherwise separate groups. HDBSCAN adjusts their distance using both endpoints' core distances:

$$d_{\text{mr}}(x, y) = \max\{\text{core}_k(x), \text{core}_k(y), d(x, y)\}$$

Pair	$\mathbf{d(x, y)}$	$\mathbf{core_k(x)}$	$\mathbf{core_k(y)}$	$\mathbf{d_{mr}(x, y)}$
Two points in dense neighborhoods	0.4	0.2	0.3	0.4
One point in a sparse neighborhood	0.4	0.2	1.1	1.1

Mutual reachability distance never decreases the original distance. It increases distances when a core distance is larger, reducing the influence of sparse bridges.

Campello, R. J., Moulavi, D., & Sander, J. (2013). **Density-based clustering based on hierarchical density estimates**. In *PAKDD* (pp. 160-172).

From adjusted distances to a hierarchy

We construct a complete weighted graph , where each point is a vertex, and each pair of points is connected by an edge weighted by their mutual reachability distance.

We then construct a minimum spanning tree (MST): a set of $n-1$ edges that connects all n points without cycles and has the smallest possible total weight.

Build the hierarchy by progressively removing the largest edges:

1. Initially, all points belong to one connected component;
2. Remove edges above the current threshold ε (maximum mutual reachability distance allowed for an edge to remain connected);
3. Each remaining connected component is a candidate group;
4. Decrease ε and record further splits.

This is the splitting view of single-linkage hierarchical clustering using mutual reachability distances.

HDBSCAN tracks persistence using: $\lambda = \frac{1}{\varepsilon}$. As λ increases, more edges disappear. Groups can lose points or split into smaller groups.

Campello, R. J., Moulavi, D., & Sander, J. (2013). **Density-based clustering based on hierarchical density estimates**. In *PAKDD* (pp. 160-172).

Condensing the hierarchy: which splits matter?

Not every split should create a new candidate cluster. A branch may simply lose a few peripheral points.

HDBSCAN uses a minimum cluster size m to distinguish a meaningful split from points leaving a continuing group.

For $m = 5$:

Split	**Interpretation**
20 points split into 12 and 8	Both groups are large enough: create two child branches
20 points split into 18 and 2	Continue the parent through the 18-point group; record the two departing points
8 points split into 4 and 4	Neither group is large enough: end the parent branch

The resulting condensed tree records:

- When each candidate branch appears.
- Which points leave it and at what λ.
- When it splits into valid children or disappears.

Two quantities have different roles:

- k, the neighborhood size, determines core distances;
- m, the minimum cluster size, determines which branches qualify as clusters.

Campello, R. J., Moulavi, D., & Sander, J. (2013). **Density-based clustering based on hierarchical density estimates**. In *PAKDD* (pp. 160-172).

Stability: which branches should we select?

A cluster receives support when many points remain in it over a broad range of λ.

For a candidate branch C, we define stability (persistence across density threshold):

$$S(C) = \sum_{x \in C} \left[\lambda_{\text{leave},C}(x) - \lambda_{\text{birth}}(C) \right]$$

- $\lambda_{birth(C)}$: when the branch appears.
- $\lambda_{(leave,C)}(x)$: when point x leaves that branch, including when the branch splits.

A parent and its descendants cannot both be selected. The comparison proceeds upward through the tree, producing a set of non-overlapping clusters; points outside the selected clusters receive the noise label.

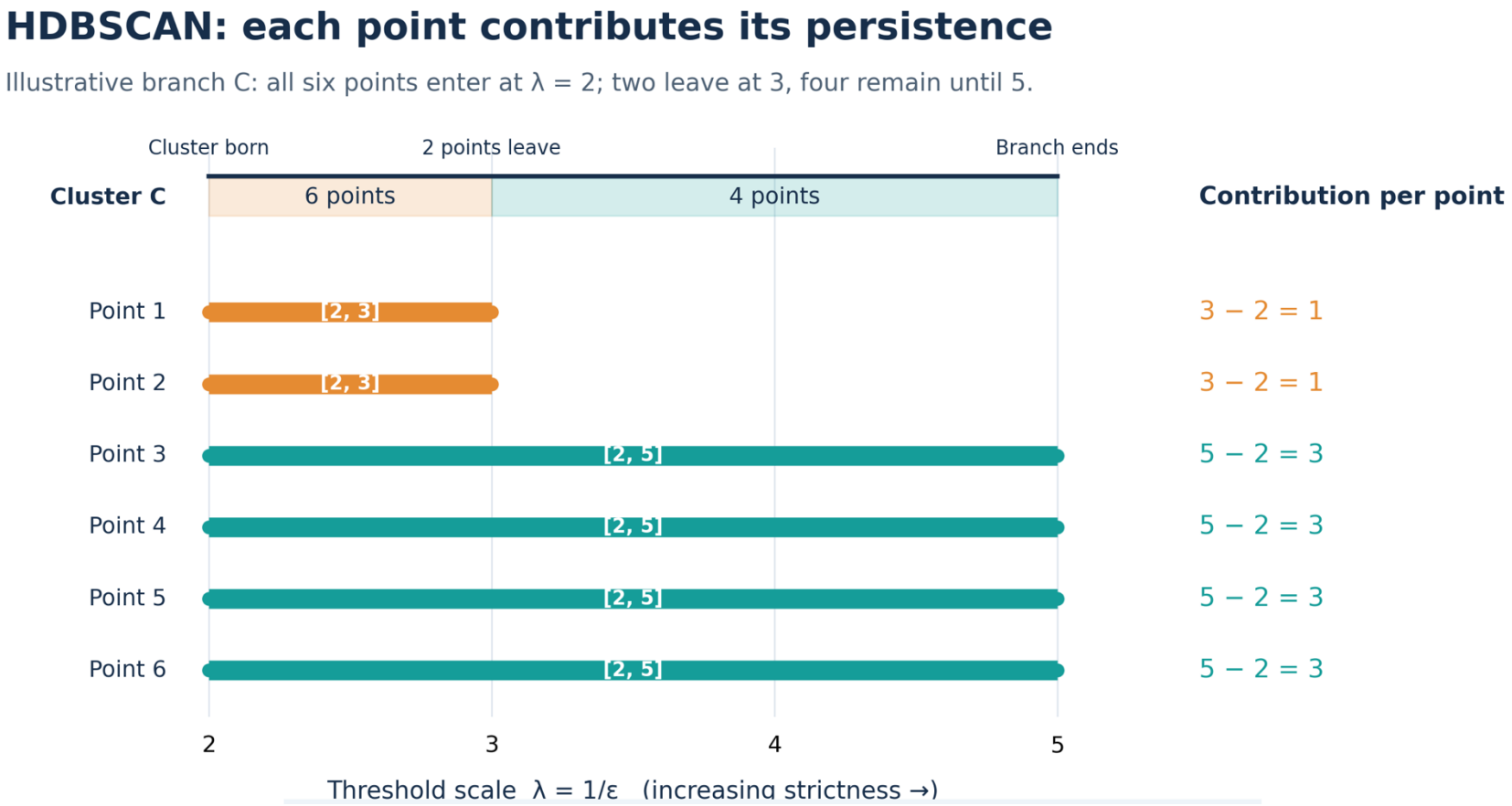
Excess of Mass: select branches by comparing their stability with the combined stability of descendants.

Campello, R. J., Moulavi, D., & Sander, J. (2013). **Density-based clustering based on hierarchical density estimates**. In *PAKDD* (pp. 160-172).

Stability: example

A branch appears at $\lambda = 2$. Two points leave at 3, and four remain until the branch ends at 5:

$$S(C) = 2(3-2) + 4(5-2) = 14$$



Campello, R. J., Moulavi, D., & Sander, J. (2013). **Density-based clustering based on hierarchical density estimates**. In *PAKDD* (pp. 160-172).

Excess of Mass: example

Assume a minimum cluster size of two points.

The parent branch C appears at $\lambda = 2$ with six points:

- Two points leave at $\lambda = 3$.
- At $\lambda = 5$, the remaining four split into two children, C_1 and C_2.
- Both children contain two points.
- C_1 persists until $\lambda = 9.5$, and C_2 until $\lambda = 9$.

Branch	**Contributions**	**Stability**
Parent C	Two points persist from 2 to 3; four from 2 to 5	$2(3-2)+4(5-2)=14$
Child C_1	Two points persist from 5 to 9.5	$2(9.5-5)=9$
Child C_2	Two points persist from 5 to 9	$2(9-5)=8$

Assuming the children have no further valid descendants, compare: $S(C) = 14$ and $S(C_1) + S(C_2) = 17$.
Select C_1 and C_2, because their combined stability is greater.

The parent and its descendants cannot both be selected.
In this example, the two points that left before the split remain noise when the children are selected.

Campello, R. J., Moulavi, D., & Sander, J. (2013). **Density-based clustering based on hierarchical density estimates**. In *PAKDD* (pp. 160-172).

From document clusters to topics

HDBSCAN assigns documents to clusters using their reduced embeddings.

A cluster identifies which documents belong together, but its numerical label does not explain their subject.

To describe each cluster, BERTopic returns to the original texts and extracts representative words.

BERTopic uses two representations for different purposes:

- Document embeddings: identify semantically related documents.
- Word frequencies: describe the content of their clusters.

The final description should emphasize words that are frequent within a cluster and informative relative to the collection.

Campello, R. J., Moulavi, D., & Sander, J. (2013). **Density-based clustering based on hierarchical density estimates**. In *PAKDD* (pp. 160-172).

BERTopic: final steps

Aggregating documents by cluster

To identify representative words, combine all documents assigned to the same cluster into one aggregated document:

$$D_c = \bigoplus_{i:y_i=c} d_i$$

where y_i is the cluster assignment of document d_i, and $\oplus$ denotes text concatenation, after tokenization and optional stopword removal, count the occurrences of each term.

Each row now represents a cluster, rather than an individual document.

This aggregation provides the input to class-based TF-IDF, where "class" refers to the cluster produced by the unsupervised algorithm.

Grootendorst, M. (2022). **BERTopic: Neural topic modeling with a class-based TF-IDF procedure**. *arXiv preprint arXiv:2203.05794*.

Aggregating documents by cluster

Term frequency alone can give high importance to words that occur frequently throughout a collection.

TF-IDF combines:

- Term frequency (TF): how frequent a word is within a document.
- Inverse document frequency (IDF): how uncommon it is across documents.

$$\text{TFIDF}(t, d) = \frac{n_{t,d}}{N_d} \qquad \log\left(\frac{M}{df_t}\right)$$

frequency within document

rarity across documents

- n_{td}: Number of occurrences of term t in document d;
- N_d: total term count in document d;
- M: number of documents;
- df_t: number of documents containing term t.

A word receives a high weight when it is frequent in a particular document but occurs in relatively few documents.

BERTopic adapts this idea to describe entire clusters.

Class-based TF-IDF

BERTopic's default c-TF-IDF assigns term t a weight in cluster c:

$$w_{t,c} = \frac{n_{t,c}}{N_c} \cdot \log\left(1 + \frac{A}{f_t}\right)$$

normalized cluster frequency

inverse total-frequency factor

- $n_{t,c}$: occurences of term t in the aggregated cluster;
- $N_c = \sum_t n_{t,c}$: total retained term count in that cluster;
- $f_t = \sum_c n_{t,c}$: total occurrences of term t across all aggregated clusters;
- A: average total term count per aggregated cluster.

The first factor normalizes for different cluster lengths. The second reduces the influence of globally frequent terms.

Unlike ordinary IDF, f_t counts occurrences, not the number of documents or clusters containing the term.

Grootendorst, M. (2022). **BERTopic: Neural topic modeling with a class-based TF-IDF procedure**. *arXiv preprint arXiv:2203.05794*.

c-TF-IDF: a numerical example

Cluster	goal	team	stock	market	today	Total
C_1	4	3	0	0	3	10
C_2	0	0	4	3	3	10
Total frequency $\mathbf{f_t}$	4	3	4	3	6	

The average cluster length is: $A = \frac{10+10}{2} = 10$.

For cluster C_1, using natural logarithm:

Term	Normalized frequency	Inverse-frequency factor	c-TF-IDF weight
goal	$4/10 = 0.4$	$\log(1 + 10/4) \approx 1.253$	0.501
team	$3/10 = 0.3$	$\log(1 + 10/3) \approx 1.466$	0.440
today	$3/10 = 0.3$	$\log(1 + 10/6) \approx 0.981$	0.294

"Team" and "today" have the same frequency within C_1, but "today" occurs more frequently across the collection and receives a lower weight.

The resulting ranking is: goal, team, today. Terms absent from C_1, such as "stock", receive weight zero.

Grootendorst, M. (2022). **BERTopic: Neural topic modeling with a class-based TF-IDF procedure**. *arXiv preprint arXiv:2203.05794*.

c-TF-IDF: a numerical example

Once c-TF-IDF has been computed, BERTopic uses it to describe the clusters that HDBSCAN has already found.

For each topic, it:

1. Ranks the terms by their c-TF-IDF scores.
2. Selects the highest-scoring terms as representative keywords.
3. Uses those keywords as the topic representation, helping you interpret its meaning.

Grootendorst, M. (2022). **BERTopic: Neural topic modeling with a class-based TF-IDF procedure**. *arXiv preprint arXiv:2203.05794*.

BERTopic: main takeways

The result depends on the whole pipeline.

- The embedding model determines which similarities are captured.
- Dimensionality reduction changes the geometry used for clustering.
- Clustering determines the document groups.
- Keyword extraction describes those groups.

A meaningful topic requires both coherent documents and informative keywords.

Evaluate the results by examining keywords, representative documents, and outliers.

A visually separated cluster alone does not establish a useful topic.

Grootendorst, M. (2022). **BERTopic: Neural topic modeling with a class-based TF-IDF procedure**. *arXiv preprint arXiv:2203.05794*.

References

1. Grootendorst, M. (2022). **BERTopic: Neural topic modeling with a class-based TF-IDF procedure**. *arXiv preprint arXiv:2203.05794*.
2. Reimers, N., & Gurevych, I. (2019, November). **Sentence-bert: Sentence embeddings using siamese bert-networks**. In *EMNLP-IJCNLP* (pp. 3982-3992).
3. McInnes, L., Healy, J., & Melville, J. (2018). **UMAP: Uniform manifold approximation and projection for dimension reduction**. *arXiv preprint arXiv:1802.03426*.
4. Campello, R. J., Moulavi, D., & Sander, J. (2013). **Density-based clustering based on hierarchical density estimates**. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 160-172).