

# **Text Clustering and Topic Modeling**

# Text clustering

Text clustering aims to group similar texts based on their semantic content, meaning, and relationships.

As illustrated in Figure 5-1, the resulting clusters of semantically similar documents not only facilitate efficient categorization of large volumes of unstructured text but also allow for quick exploratory data analysis.

Language is more than a bag of words, and recent language models have proved to be quite capable of capturing that notion.

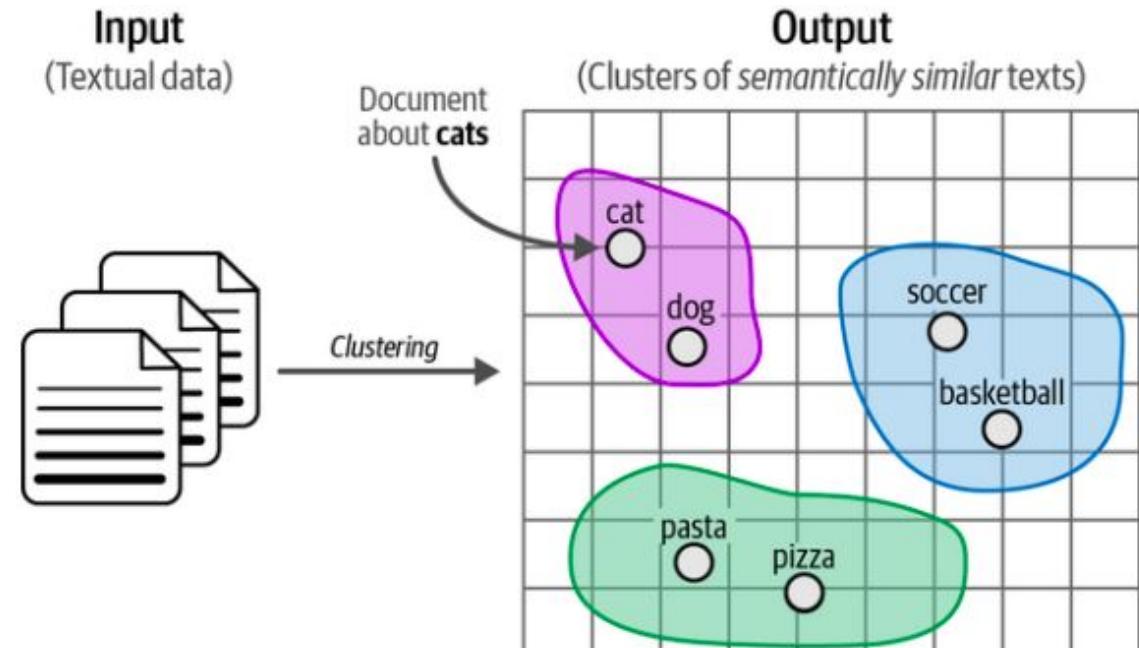
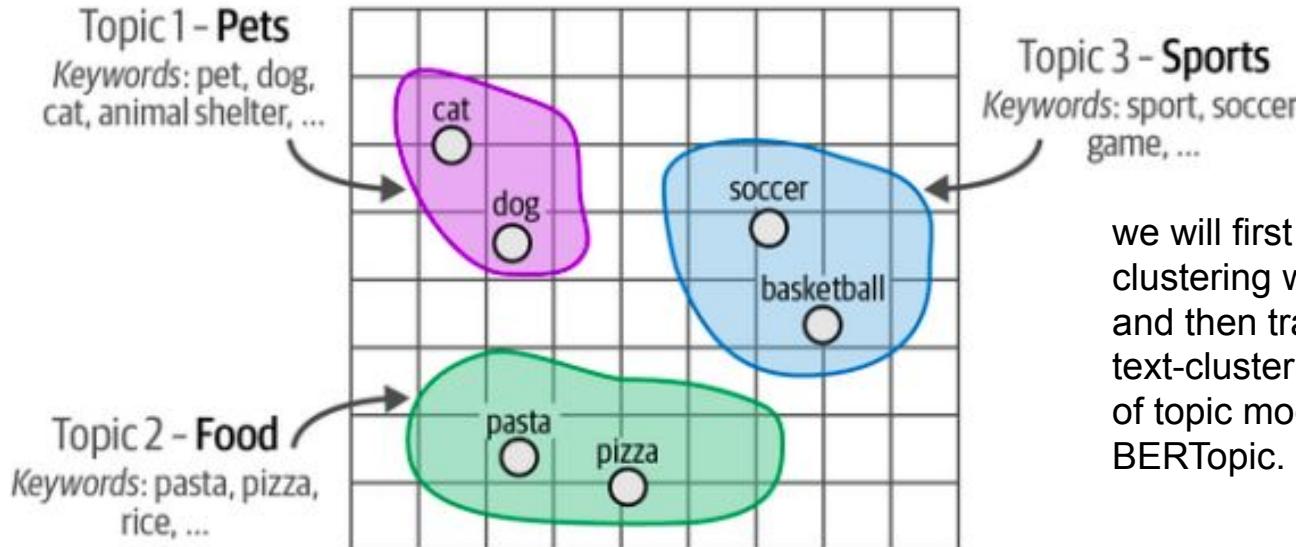


Figure 5-1. Clustering unstructured textual data.

# Topic Modelling

Text clustering has also found itself in the realm of topic modeling, where we want to discover (abstract) topics that appear in large collections of textual data. As shown in Figure 5-2, we generally describe a topic using keywords or keyphrases and, ideally, have a single overarching label.



we will first explore how to perform clustering with embedding models and then transition to a text-clustering-inspired method of topic modeling, namely BERTopic.

*Figure 5-2. Topic modeling is a way to give meaning to clusters of textual documents.*

# A Common Pipeline for Text Clustering

Although there are many methods for text clustering, from graph-based neural networks to centroid-based clustering techniques, a common pipeline that has gained popularity involves three steps and algorithms:

1. Convert the input documents to embeddings with an embedding model.
2. Reduce the dimensionality of embeddings with a dimensionality reduction model.
3. Find groups of semantically similar documents with a cluster model.

# Step 1 - Embedding Documents

The first step is to convert our textual data to embeddings, as illustrated in Figure 5-3. Recall from previous chapters that embeddings are numerical representations of text that attempt to capture its meaning.

most embedding models at the time of writing focus on just that, semantic similarity.

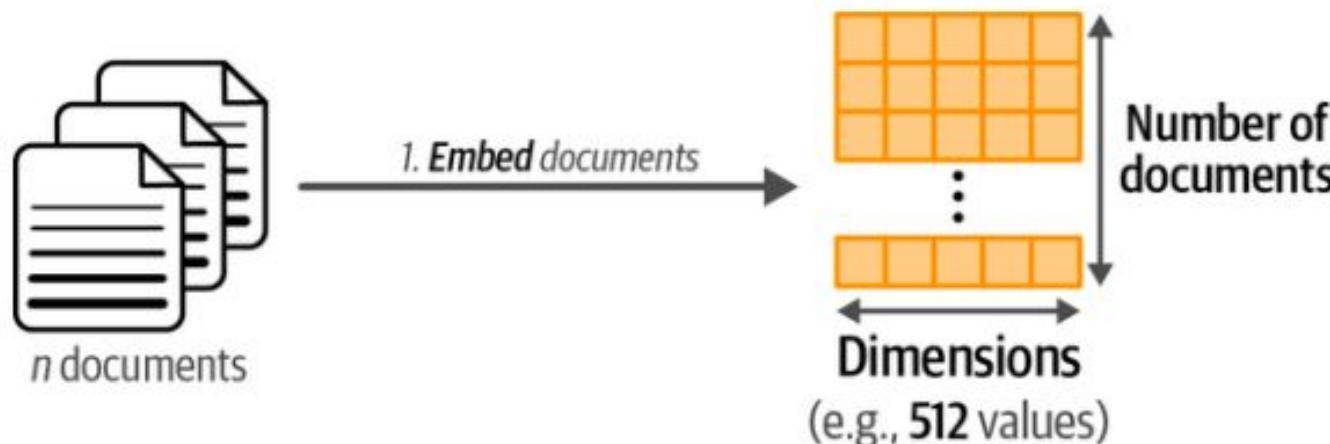


Figure 5-3. Step 1: We convert documents to embeddings using an embedding model.

the “[thenlper/gte-small](#)” model instead. It is a more recent model that outperforms the previous model on clustering tasks and due to its small size is even faster for inference.

# Reducing the Dimensionality of Embeddings

As the number of dimensions increases, there is an exponential growth in the number of possible values within each dimension. Finding all subspaces within each dimension becomes increasingly complex.

we can make use of dimensionality reduction. As illustrated in Figure 5-4, this technique allows us to reduce the size of the dimensional space and represent the same data with fewer dimensions.

Dimensionality reduction techniques aim to preserve the global structure of high-dimensional data by finding low-dimensional representations.

# Step 2 - Reducing the Dimensionality of Data

## Dimension Reduction

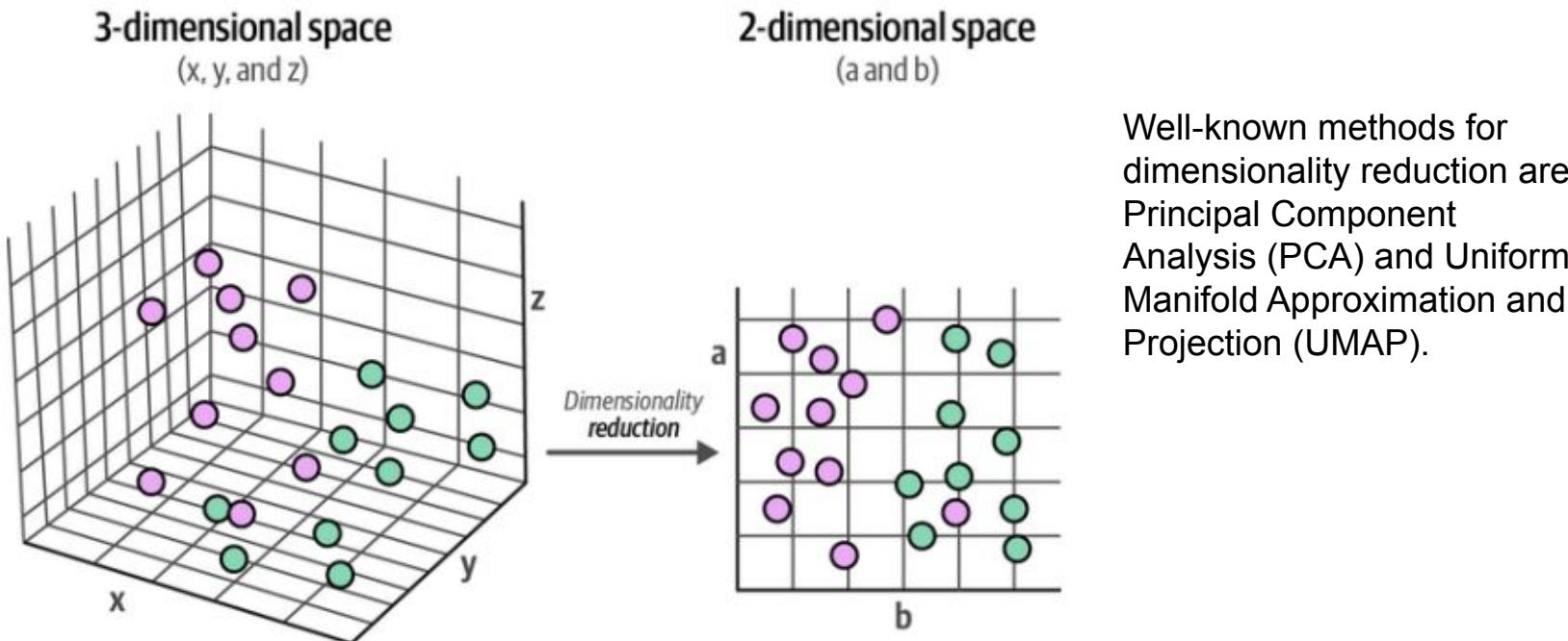
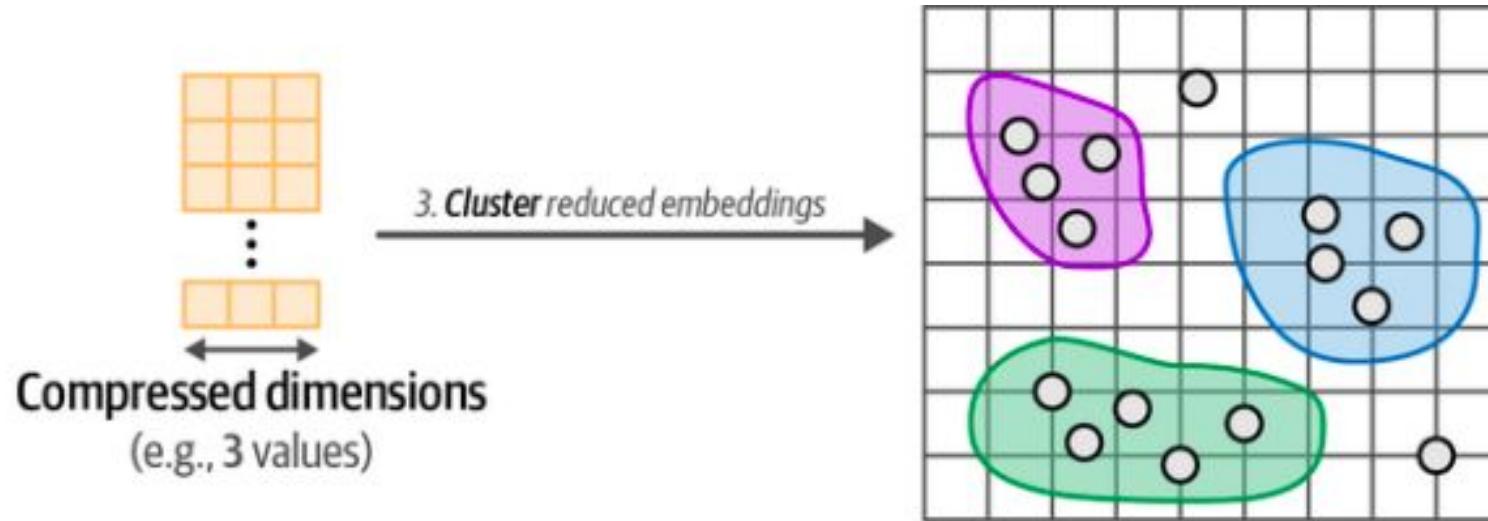


Figure 5-4. Dimensionality reduction allows data in high-dimensional space to be compressed to a lower-dimensional representation.

Well-known methods for dimensionality reduction are Principal Component Analysis (PCA) and Uniform Manifold Approximation and Projection (UMAP).

# Step 3 - Cluster the Reduced Embeddings

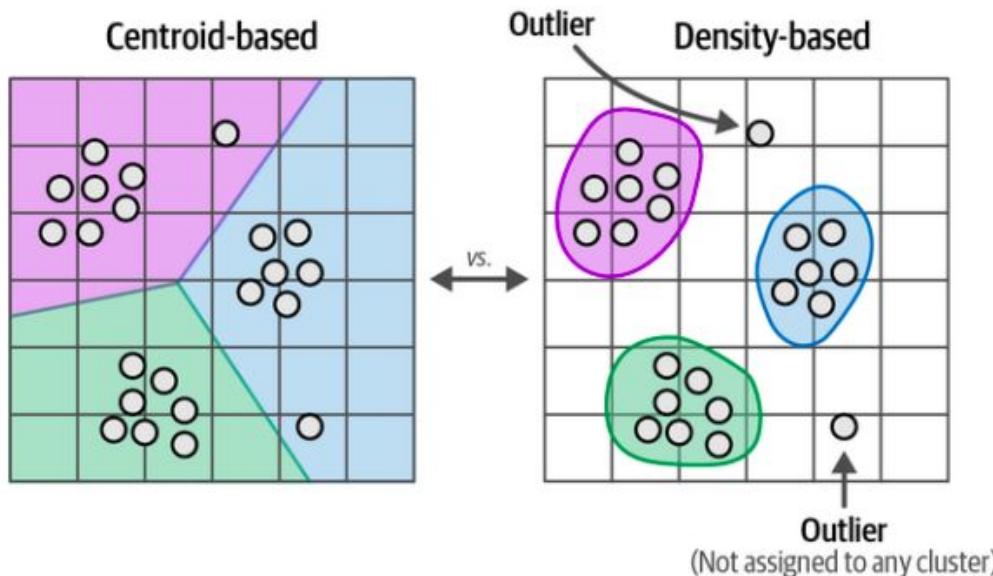
The third step is to cluster the reduced embeddings, as illustrated in Figure 5-6.



*Figure 5-6. Step 3: We cluster the documents using the embeddings with reduced dimensionality.*

# Step 3 - Cluster the Reduced Embeddings

Although a common choice is a centroid-based algorithm like k-means, which requires a set of clusters to be generated, we do not know the number of clusters beforehand. Instead, a density-based algorithm freely calculates the number of clusters and does not force all data points to be part of a cluster, as illustrated in Figure 5-7.



As a density-based method, HDBSCAN can also detect outliers in the data, which are data points that do not belong to any cluster. These outliers will not be assigned or forced to belong to any cluster.

Figure 5-7. The clustering algorithm not only impacts how clusters are generated but also how they are viewed.

# From Text Clustering to Topic Modeling

This idea of finding themes or latent topics in a collection of textual data is often referred to as topic modeling. Traditionally, it involves finding a set of keywords or phrases that best represent and capture the meaning of the topic, as we illustrate in Figure 5-9.

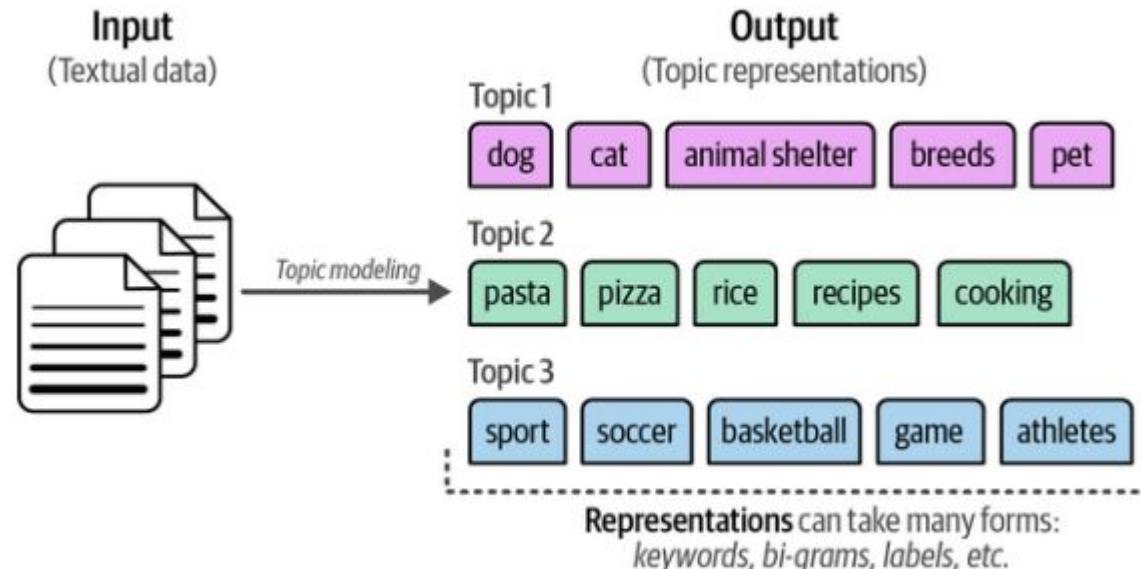


Figure 5-9. Traditionally, topics are represented by a number of keywords but can take other forms.

# Traditional Dirichlet technology

Classic approaches, like latent Dirichlet allocation, assume that each topic is characterized by a probability distribution of words in a corpus's Vocabulary. Figure 5-10 demonstrates how each word in a vocabulary is scored against its relevance to each topic.



Figure 5-10. Keywords are extracted based on their distribution over a single topic.

# BERTopic: A Modular Topic Modeling Framework

There are two steps in the Pipeline for BERTopic. First Step, as illustrated in Figure 5-11, we follow the same procedure as we did before in our text clustering example. We embed documents, reduce their dimensionality, and finally cluster the reduced embedding to create groups of semantically similar documents.

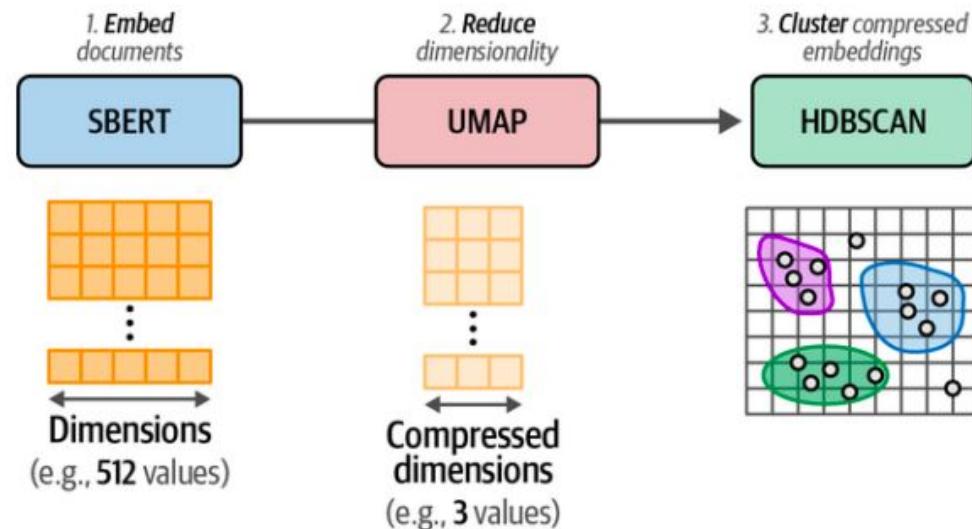


Figure 5-11. The first part of BERTopic's pipeline is to create clusters of semantically similar documents.

# BERTopic: A Modular Topic Modeling Framework

Second, it models a distribution over words in the corpus's vocabulary by leveraging a classic method, namely bag-of-words. The bag-of-words, as we discussed briefly in Chapter 1 and illustrate in Figure 5-12, does exactly what its name implies, counting the number of times each word appears in a document. The resulting representation could be used to extract the most frequent words inside a document.

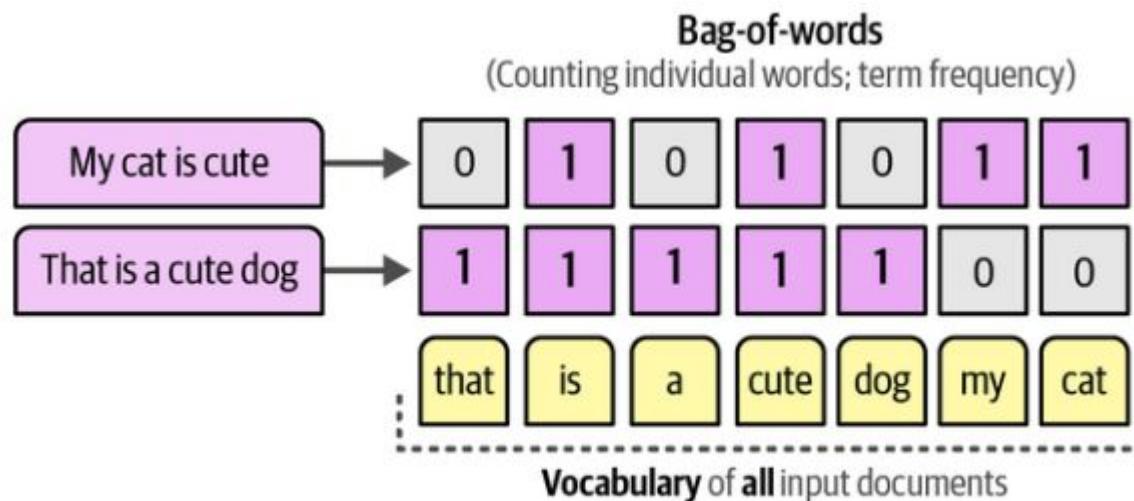


Figure 5-12. A bag-of-words counts the number of times each word appears inside a document.

# From TF to c-TF

First, this is a representation on a document level and we are interested in a cluster-level perspective. To address this, the frequency of words is calculated within the entire cluster instead of only the document, as illustrated in Figure 5-13.

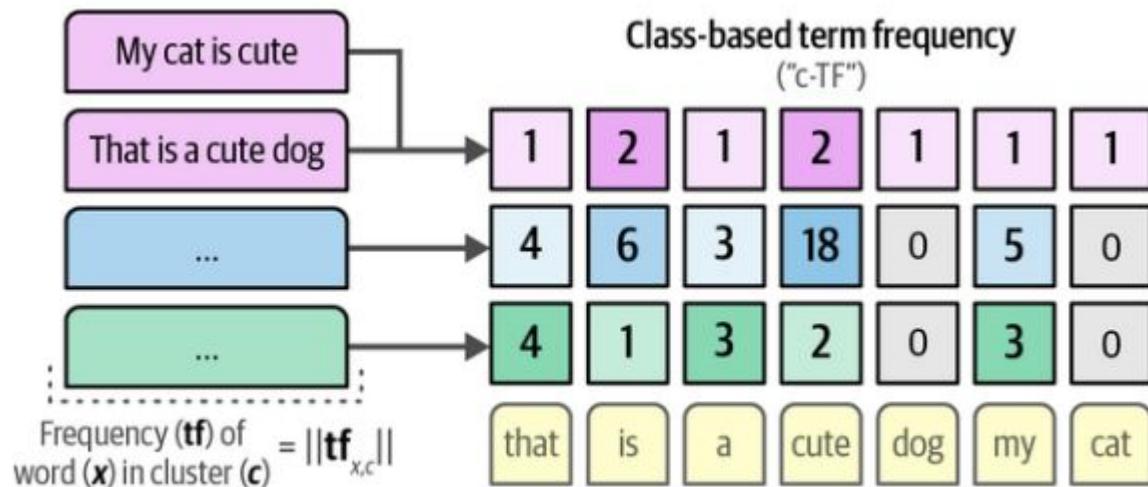


Figure 5-13. Generating c-TF by counting the frequency of words per cluster instead of per document.

# Inverse Document Frequency - IDF

Second, stop words like “the” and “I” tend to appear often in documents and provide little meaning to the actual documents. BERTopic uses a class-based variant of term frequency-inverse document frequency (c-TF-IDF) to put more weight on words that are more meaningful to a cluster and put less weight on words that are used across all clusters.

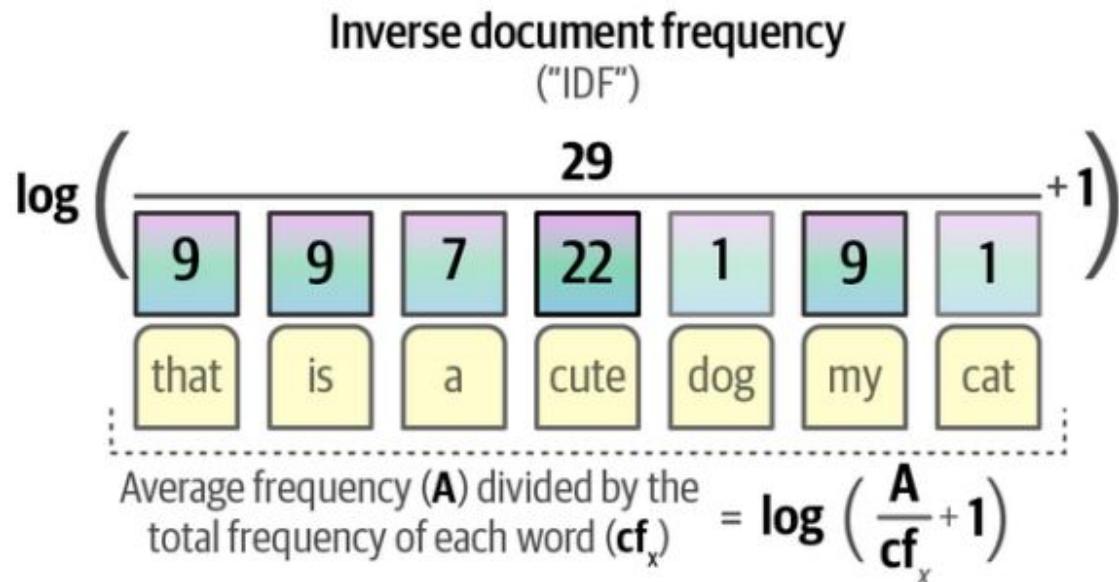


Figure 5-14. Creating a weighting scheme.

# The calculation of the weight of term $*x*$ in a class $*c*$ .

This second part of the procedure, as shown in Figure 5-15, allows for generating a distribution over words as we have seen before. We can use scikit-learn's CountVectorizer to generate the bag-of-words (or term frequency) representation. Here, each cluster is considered a topic that has a specific ranking of the corpus's vocabulary.

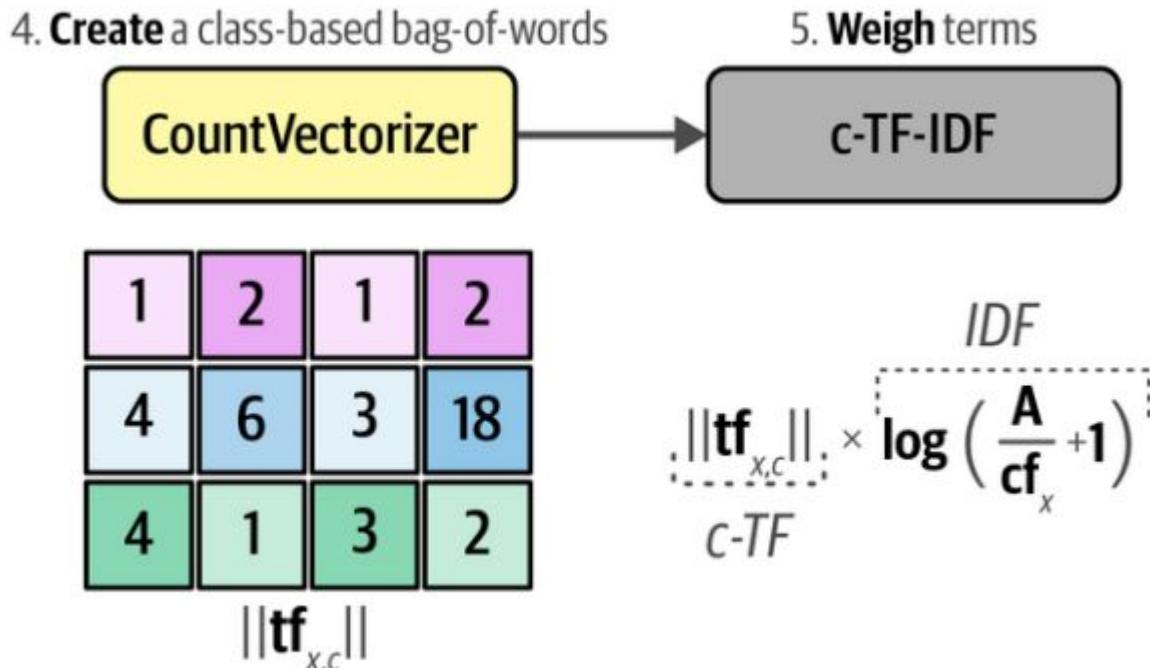
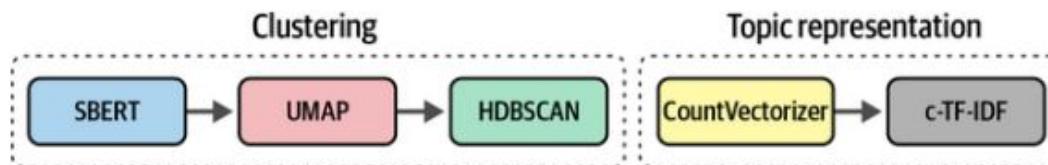


Figure 5-15. The second part of BERTopic's pipeline is representing the topics: the calculation of the weight of term  $*x*$  in a class  $*c*$ .

# Final Pipeline for BERTopic

A major advantage of this pipeline is that the two steps, clustering and topic representation, are largely independent of one another. For instance, with c-TF-IDF, we are not dependent on the models used in clustering the documents. This allows for significant modularity throughout every component of the pipeline.

Putting the two steps together, clustering and representing topics, results in the full pipeline of BERTopic, as illustrated in [Figure 5-16](#). With this pipeline, we can cluster semantically similar documents and from the clusters generate topics represented by several keywords. The higher a word's weight in a topic, the more representative it is of that topic.



*Figure 5-16. The full pipeline of BERTopic, roughly, consists of two steps, clustering and topic representation.*

# Using Generative Models for Text Clustering

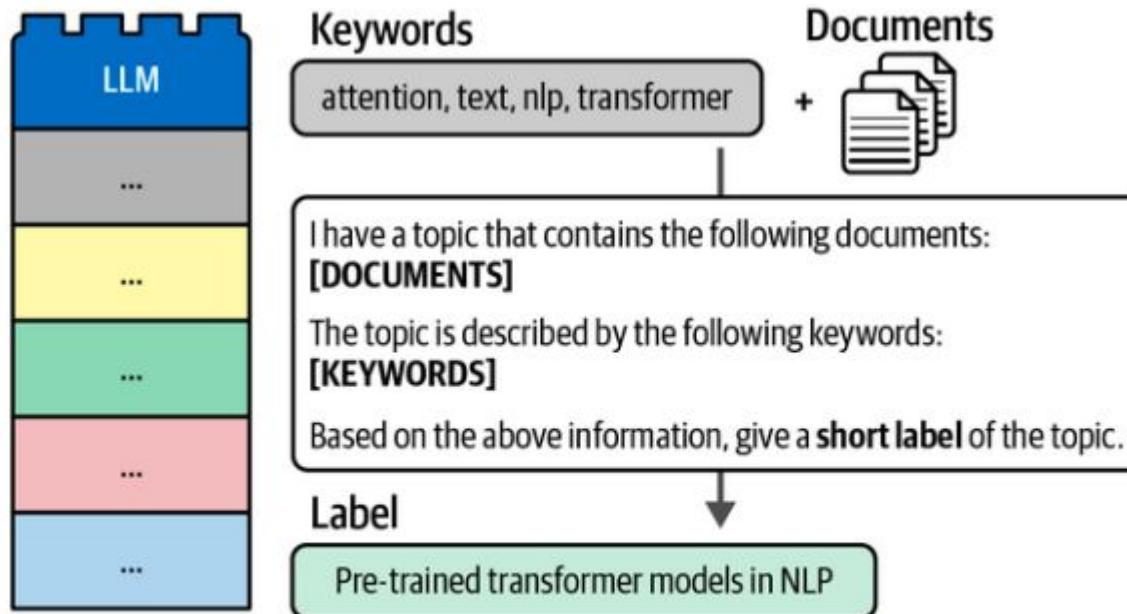


Figure 5-23. Use text generative LLMs and prompt engineering to create labels for topics from keywords and documents related to each topic.