Clustering with DBSCAN

Niklas Greifenstein, Marion Thaler

December 16, 2024

Outline

Clustering

"Clustering algorithms group a nset [of objects] into subsets or clusters. The algorithms' goal is to create clusters that are coherent internally, but clearly different from each other." (Schütze et al., 2008)

- unsupervised learning
- exploratory

Clustering Types

- Partitioning Clustering
  - **divides** data into non-overlapping clusters
  - k-Means
- Hierarchical Clustering
  - builds a **tree-like** structure through merging or splitting
  - single linkage, Divisive Analysis Clustering (DIANA)
- Density-Based Clustering
  - identifies clusters as **dense regions**
  - DBSCAN, OPTICS
- Neural Networks
  - combines clustering with **deep learning**
  - Deep embedded clustering (DEC)
- ...

Evaluation

- Exploration as goal: Evaluation metric depends on use-case
- Two types of metrics:
  - **Internal** evaluation
    - similarity-based
    - Silhouette, Davies-Bouldin, ...
  - **External** evaluation
    - label-based (requires ground truth, typically not available in clustering scenarios)
    - Adjusted Rand Index, Normalized Mutual Information, Purity, ...

Basic Idea

Clusters are dense regions in the data space, separated by regions of lower density.

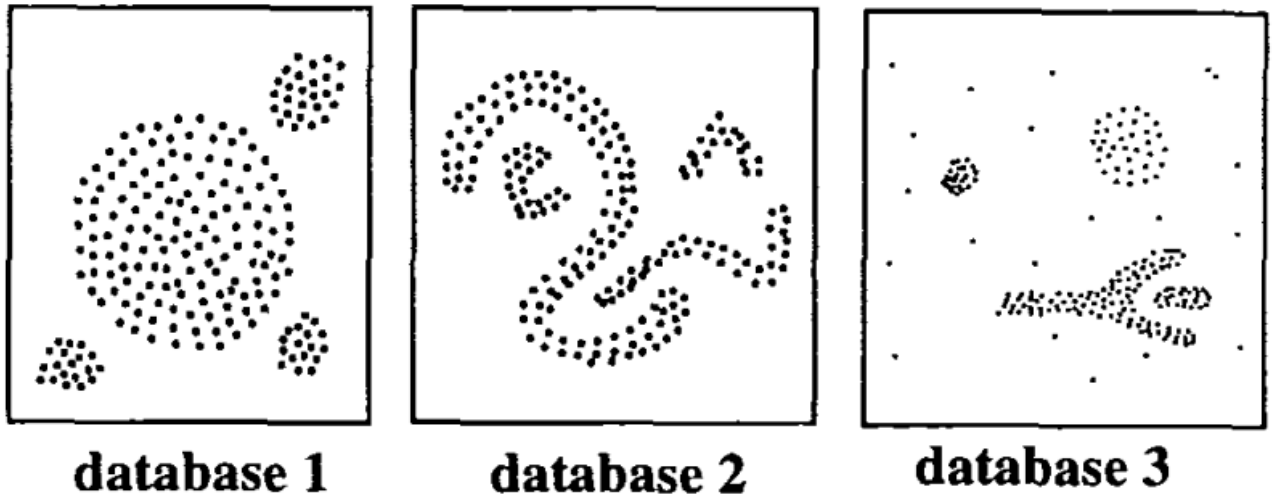


Figure: Ester et al. (1996)

Local Point Density

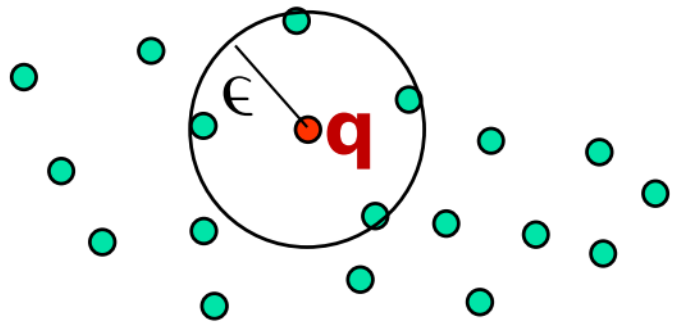


Figure: Ester (2018)

Local point density at a point q is defined by two parameters:

- ϵ-radius for the neighborhood of point q:

$$N_\epsilon(q) = \{p \in D \mid \text{dist}(p, q) \leq \epsilon\} \tag{1}$$

- **MinPts**: the minimum number of points in the given neighborhood $N_\epsilon(q)$.

Core vs Border Point

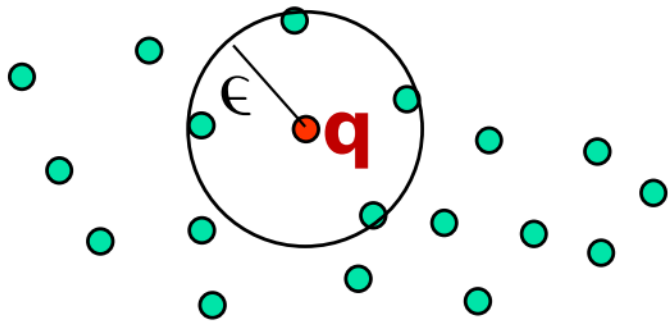


Figure: Ester (2018)

Point $q \in D$ is a **core point** (w.r.t D) if:

$$|N_\epsilon(q)| \geq \text{MinPts}, \quad \text{with } N_\epsilon(q) = \{q' \in D \mid \text{dist}(q, q') \leq \epsilon\}. \tag{2}$$

Point $p \in D$ is a **border point** (w.r.t D) if it is not a core point but is density-reachable from another core point.

(Directly) Density-Reachable

Figure: Ester (2018)

Point p is **directly density-reachable** from q w.r.t. ϵ, MinPts if:

- $p \in N_\epsilon(q)$, and
- q is a core point w.r.t. ϵ, MinPts.

Point p is **density-reachable** from q, if there is a chain of directly density reachable objects between q and p.

Density-Connected

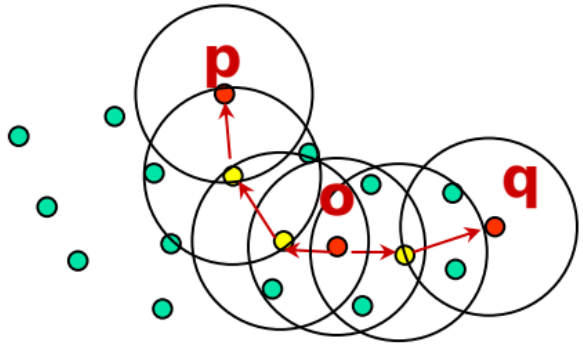


Figure: Ester (2018)

Point p is **density-connected** to a point q w.r.t. ϵ, MinPts if there is a point o such that both p and q are density-reachable from o w.r.t. ϵ, MinPts.

Density-based Cluster

Cluster C w.r.t. ϵ and MinPts is a non-empty subset of D satisfying:

- **Maximality**: $\forall p, q \in D$, if $p \in C$ and q is density-reachable from p, then $q \in C$.
- **Connectivity**: $\forall p, q \in C$, p is density-connected to q.

Outline

DBSCAN

- **D**ensity-**B**ased **S**pacial **C**lustering for **A**pplications with **N**oise
- Developed in 1996 at LMU! (Ester et al., 1996)

- Idea: Create a clustering algorithm that:
  - does not require domain knowledge
  - can deal with higher dimensions and arbitrary shapes
  - can detect outliers

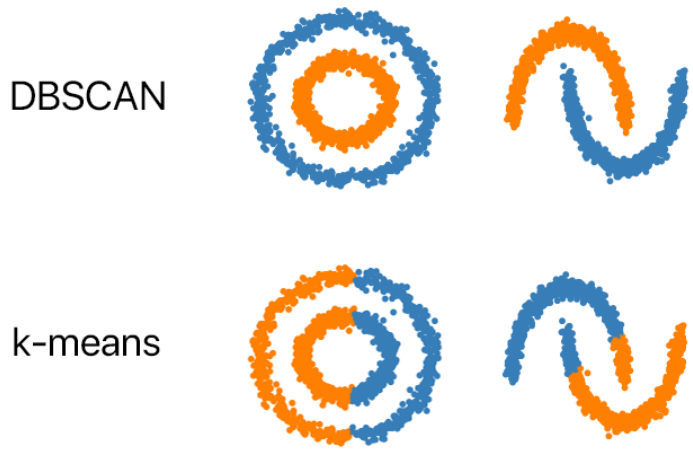


Figure: Pedregosa et al. (2011)

How does it work?

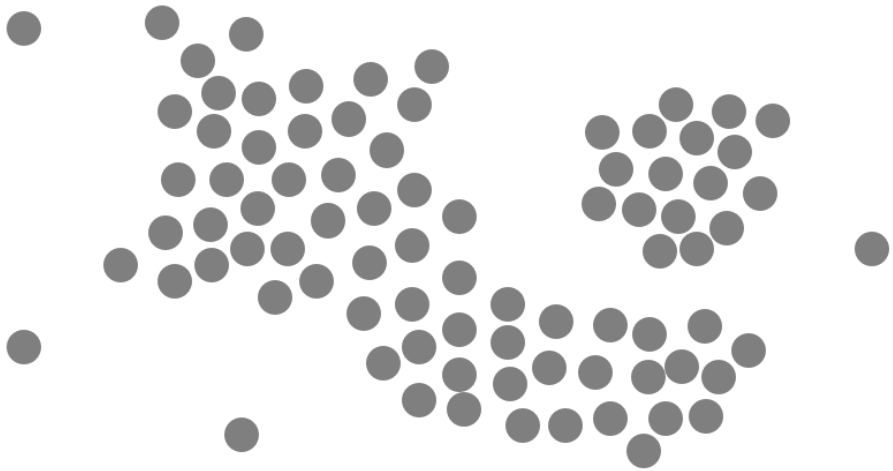
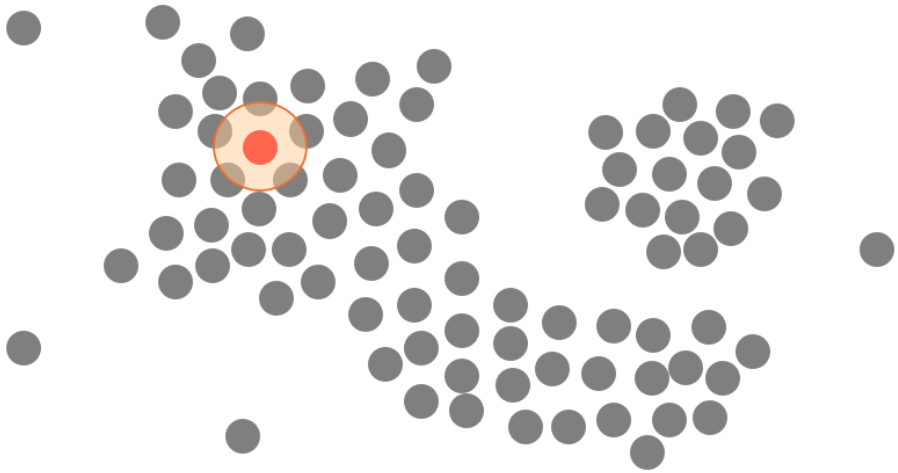


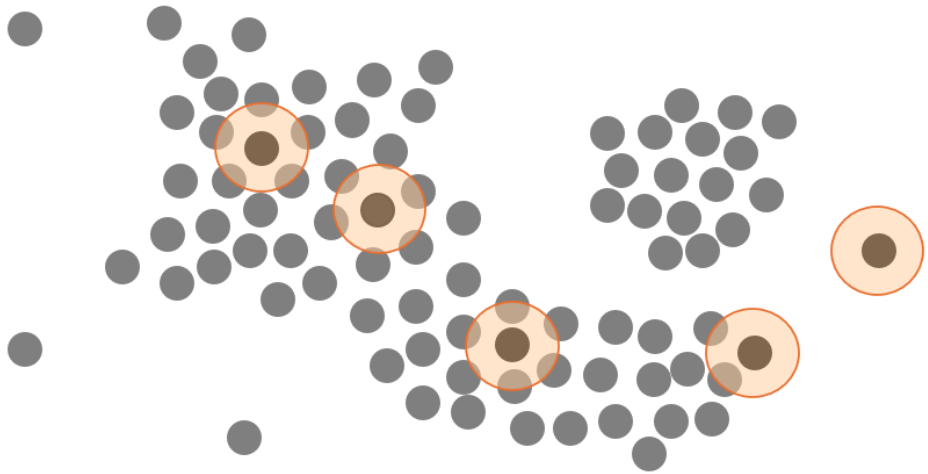
Figure: Visuals inspired by Starmer (2022)

Compute neighbors of each point



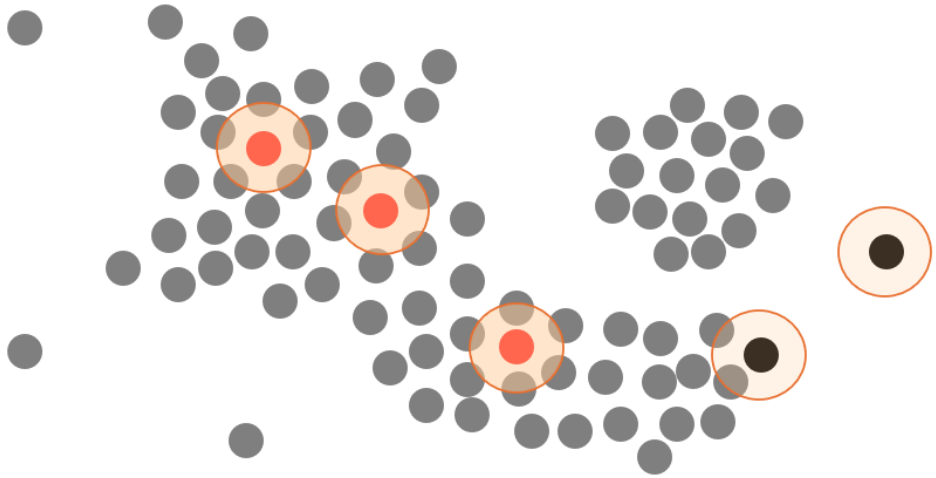
- 1. hyper-parameter: ε (circle radius)
- 2. hyper-parameter: distance metric (e.g., $L1$, cosine similarity . . .)

Identify core points

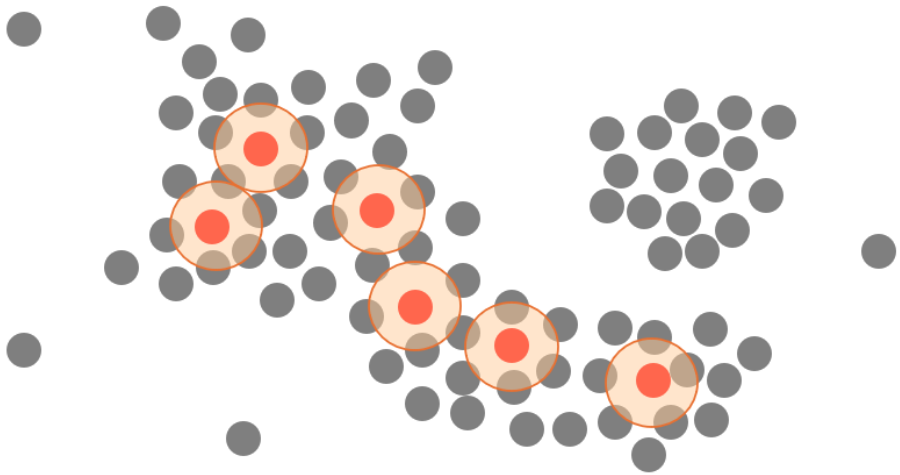


- 3. hyper-parameter: *MinPts* (threshold for the number of neighbors)

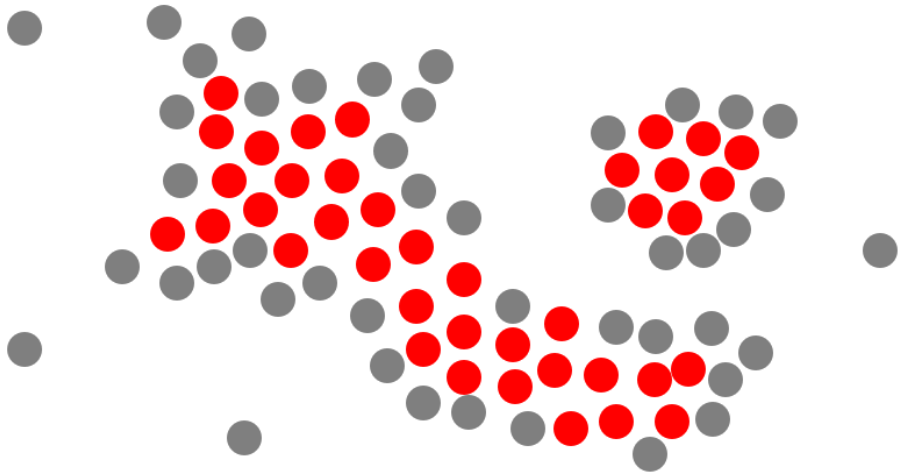
Identify core points



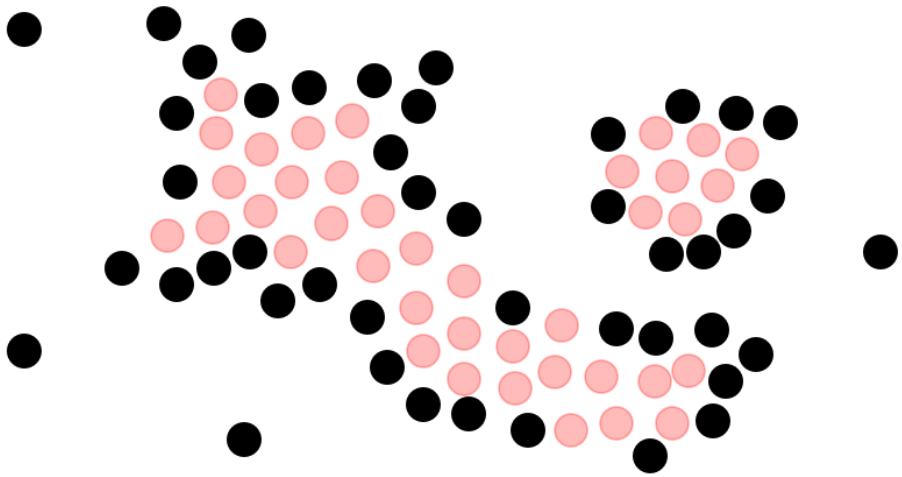
Identify core points



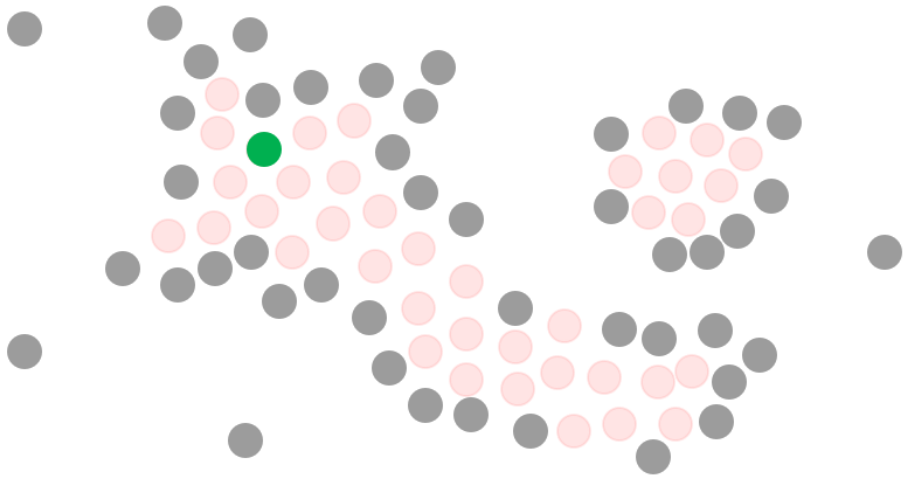
Identify core points



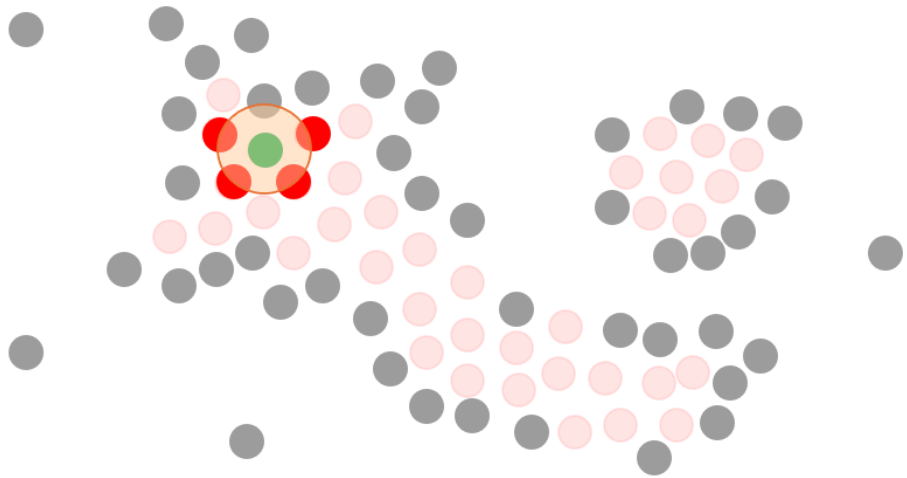
Border points



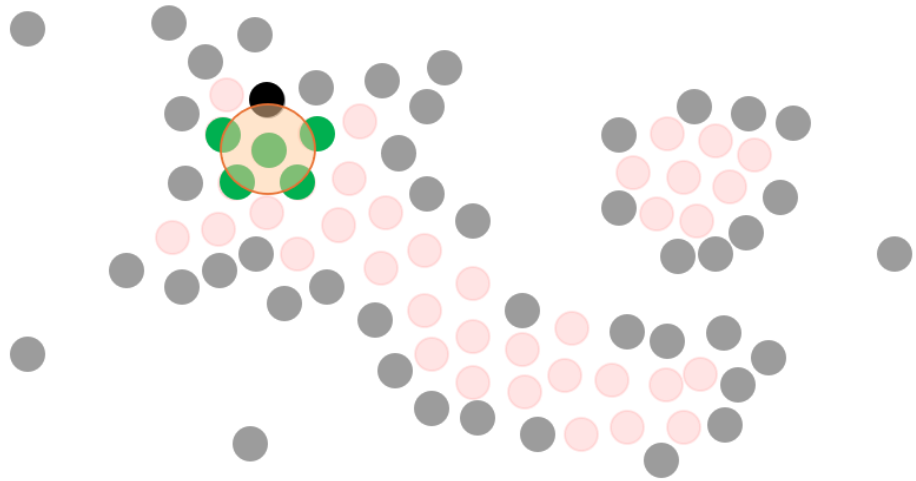
Join neighboring core points into clusters



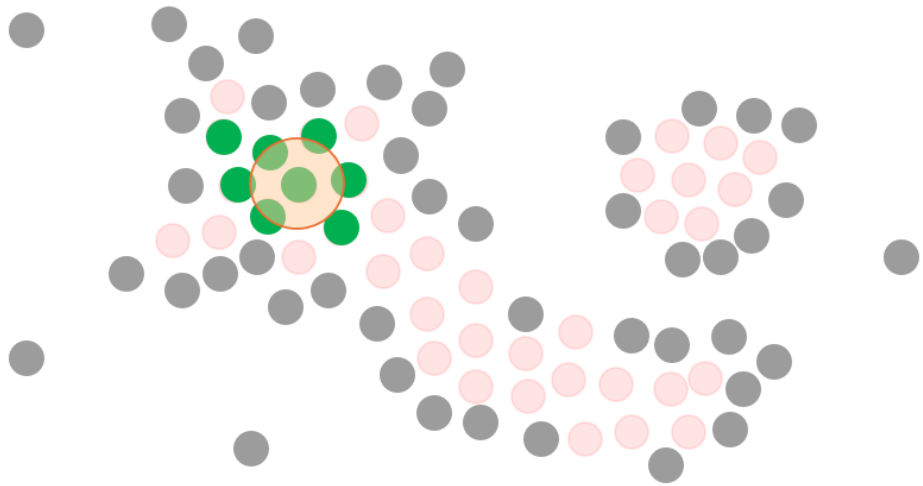
Join neighboring core points into clusters



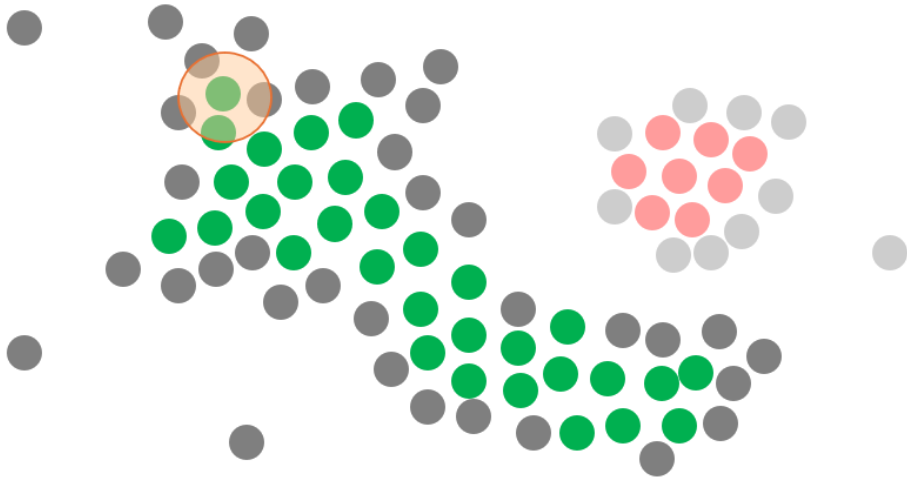
Join neighboring core points into clusters



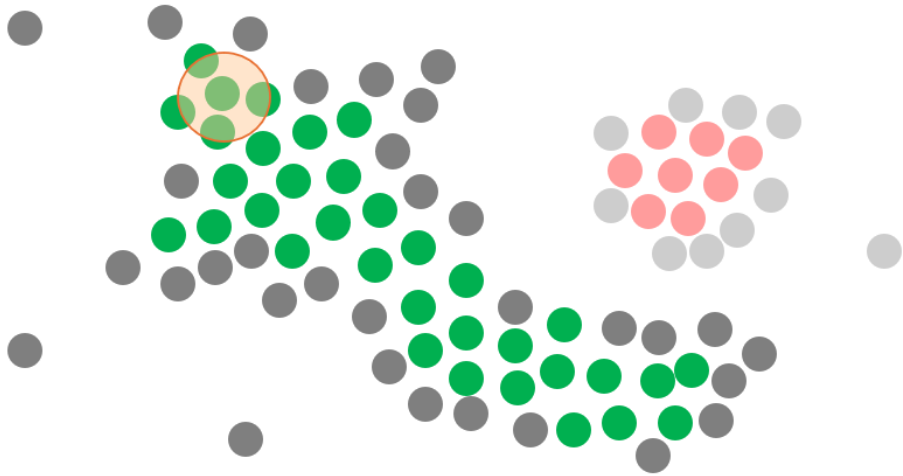
Join neighboring core points into clusters



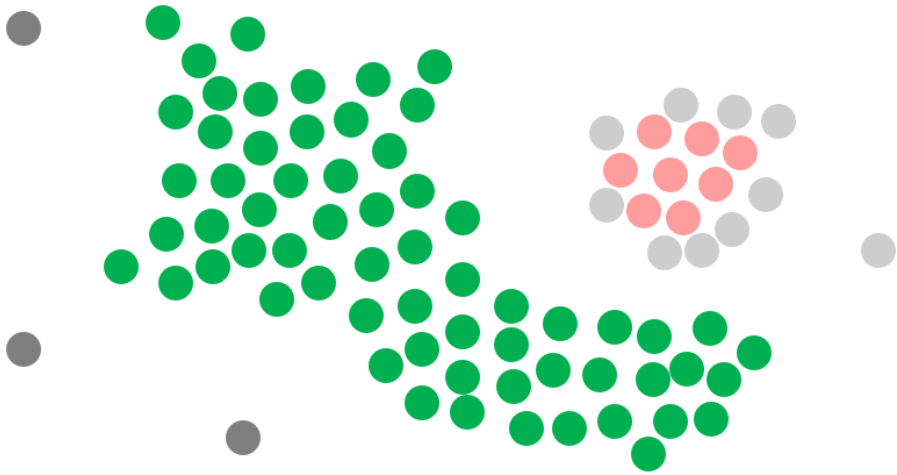
Add border point to a neighboring core point



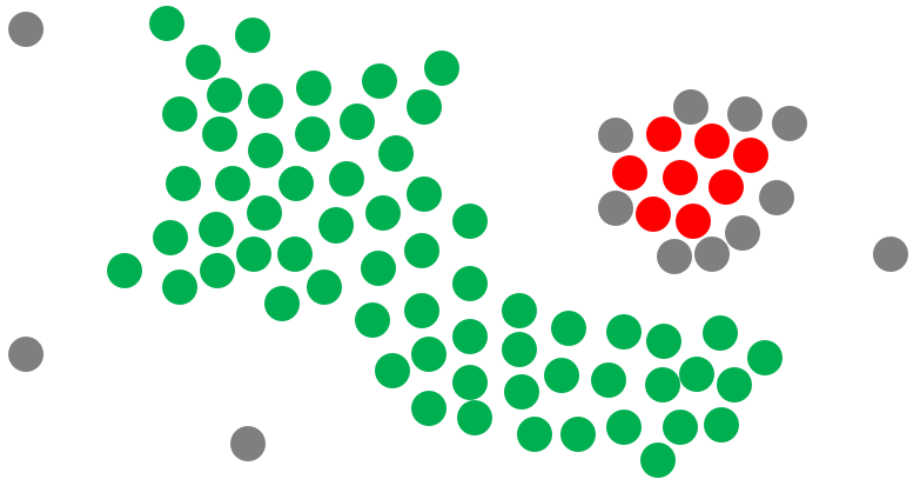
Add border points without neighbor to noise



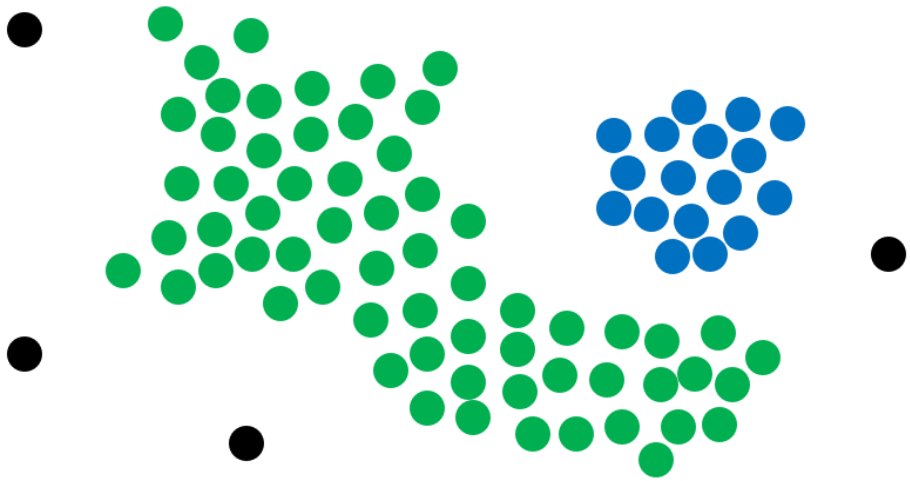
Join neighboring core points into clusters



Clustering finished



Clustering finished



Algorithm

ALGORITHM 1: Pseudocode of Original Sequential DBSCAN Algorithm

Input: *DB*: Database
Input: ε: Radius
Input: *minPts*: Density threshold
Input: *dist*: Distance function
Data: *label*: Point labels, initially *undefined*

```
1  foreach point p in database DB do                          // Iterate over every point
2      if label(p) ≠ undefined then continue                  // Skip processed points
3      Neighbors N ← RangeQuery(DB, dist, p, ε)               // Find initial neighbors
4      if |N| < minPts then                                   // Non-core points are noise
5          label(p) ← Noise
6          continue
7      c ← next cluster label                                 // Start a new cluster
8      label(p) ← c
9      Seed set S ← N \ {p}                                   // Expand neighborhood
10     foreach q in S do
11         if label(q) = Noise then label(q) ← c
12         if label(q) ≠ undefined then continue
13         Neighbors N ← RangeQuery(DB, dist, q, ε)
14         label(q) ← c
15         if |N| < minPts then continue                      // Core-point check
16         S ← S ∪ N
```

Figure: Schubert et al. (2017)

Parameter Choice

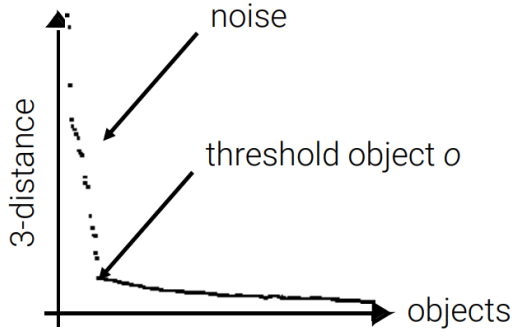


Figure: Behrisch (2021)

- **k-distance**: distance from p to its k-nearest neighbor.
- **set MinPts** (default: $2d - 1$, where d is the data space dimension (Sander et al., 1998))
- **compute k-distance**: for all $p \in D$, calculate k-distance ($k = \text{MinPts}$) and plot in descending order.
- **select** ϵ: choose border point o from the plot; set ϵ to the MinPts-distance of o.

Parameter Choice

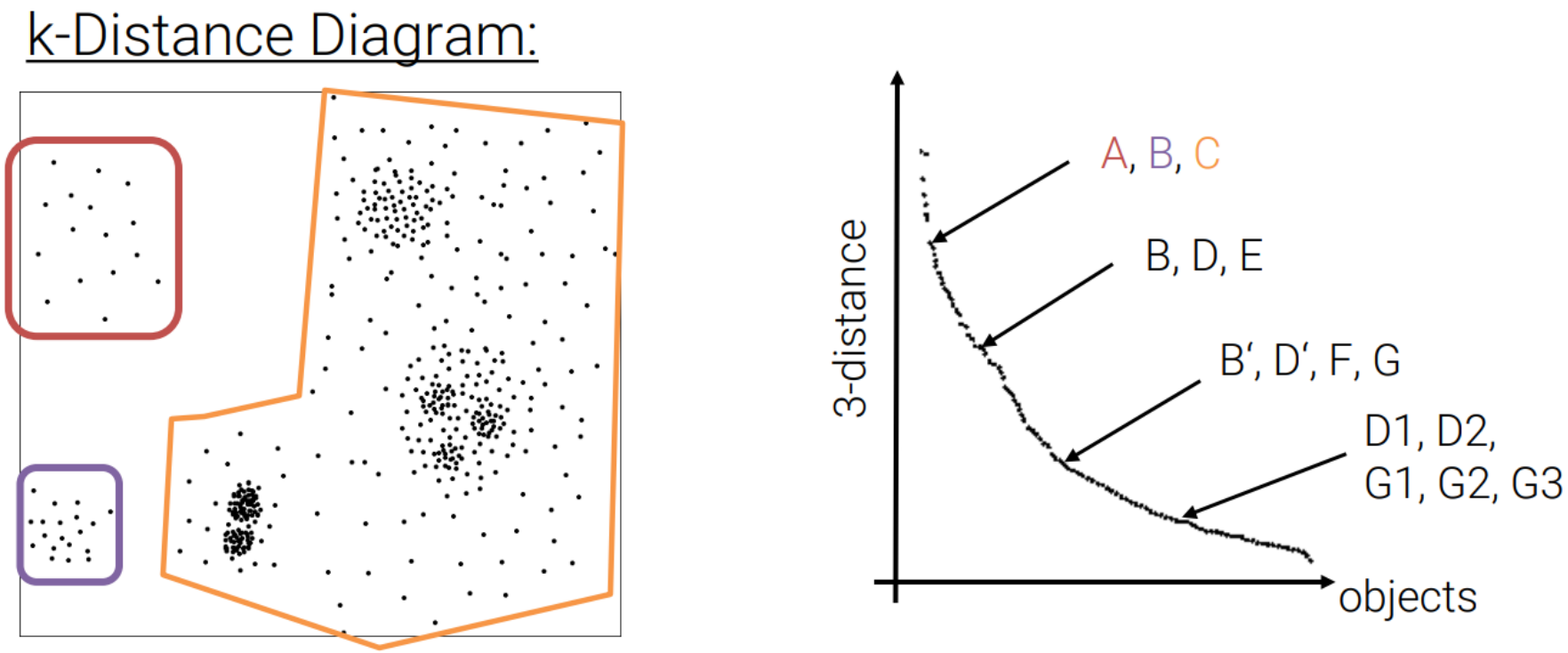


Figure: Behrisch (2021)

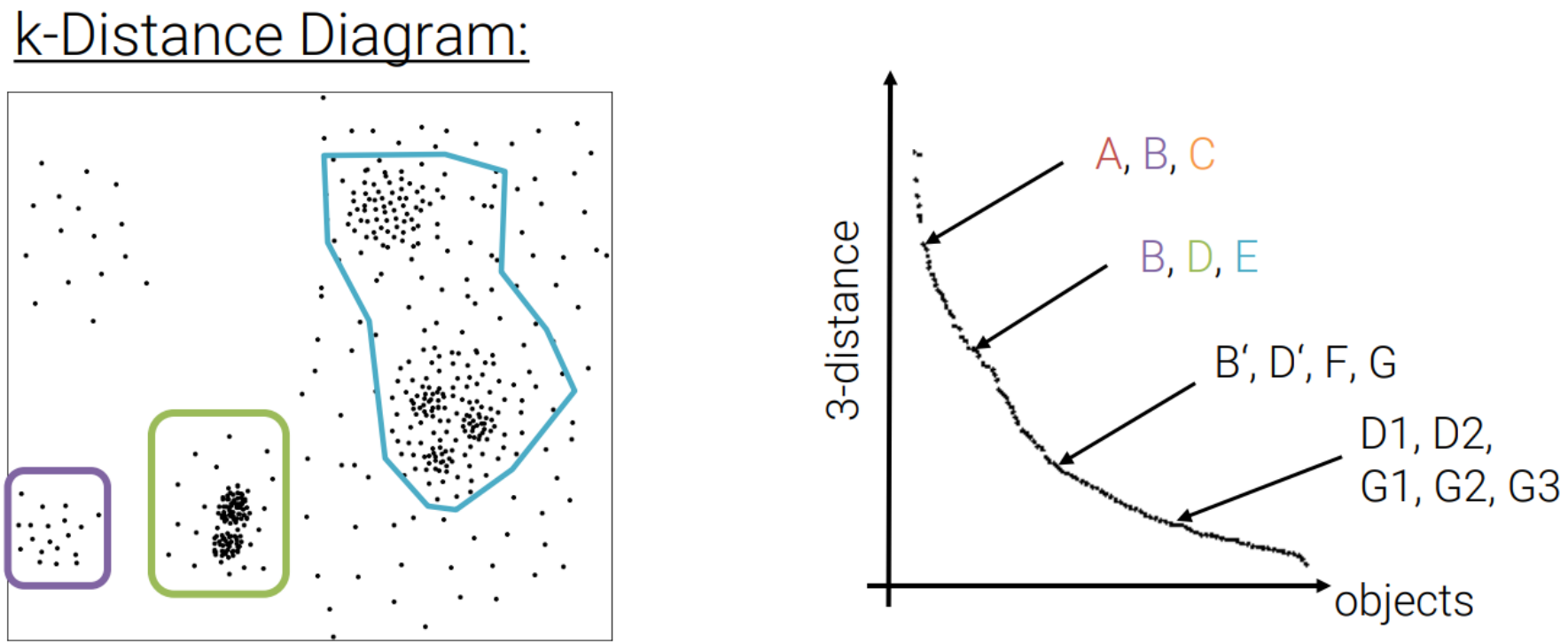


Figure: Behrisch (2021)

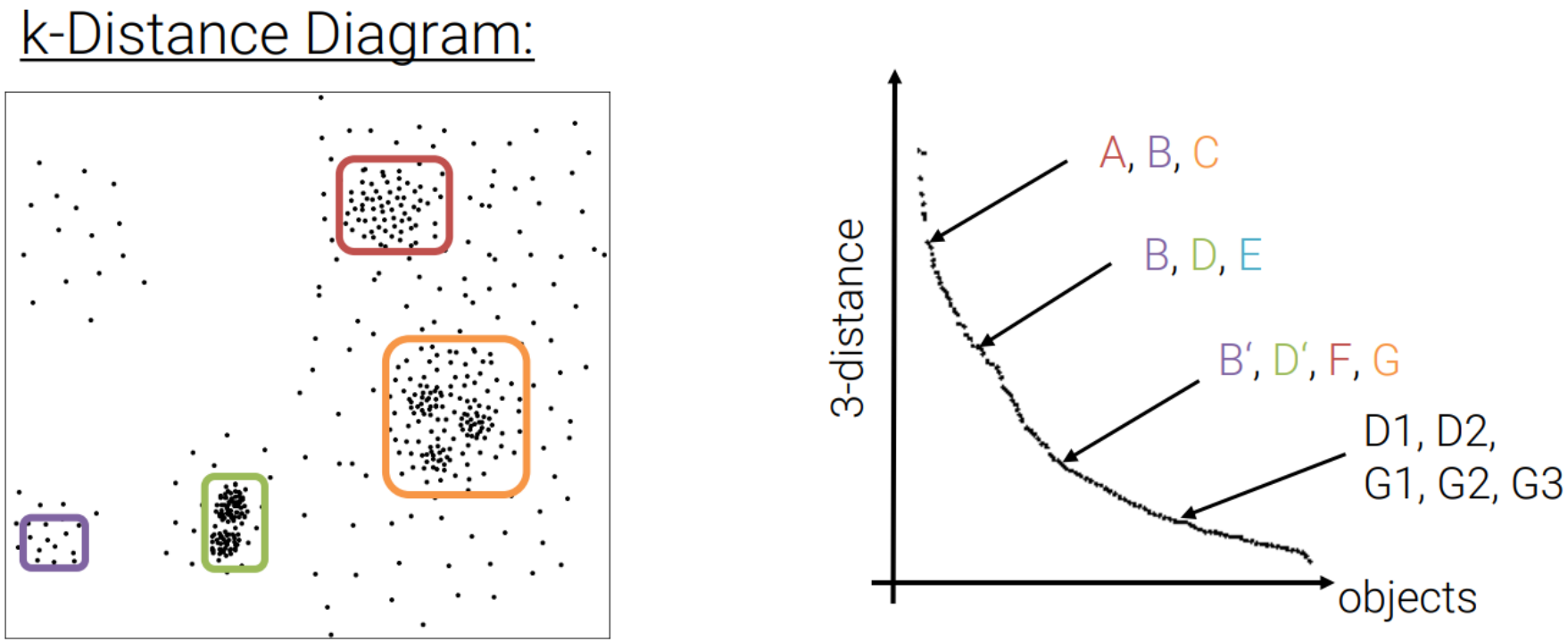


Figure: Behrisch (2021)

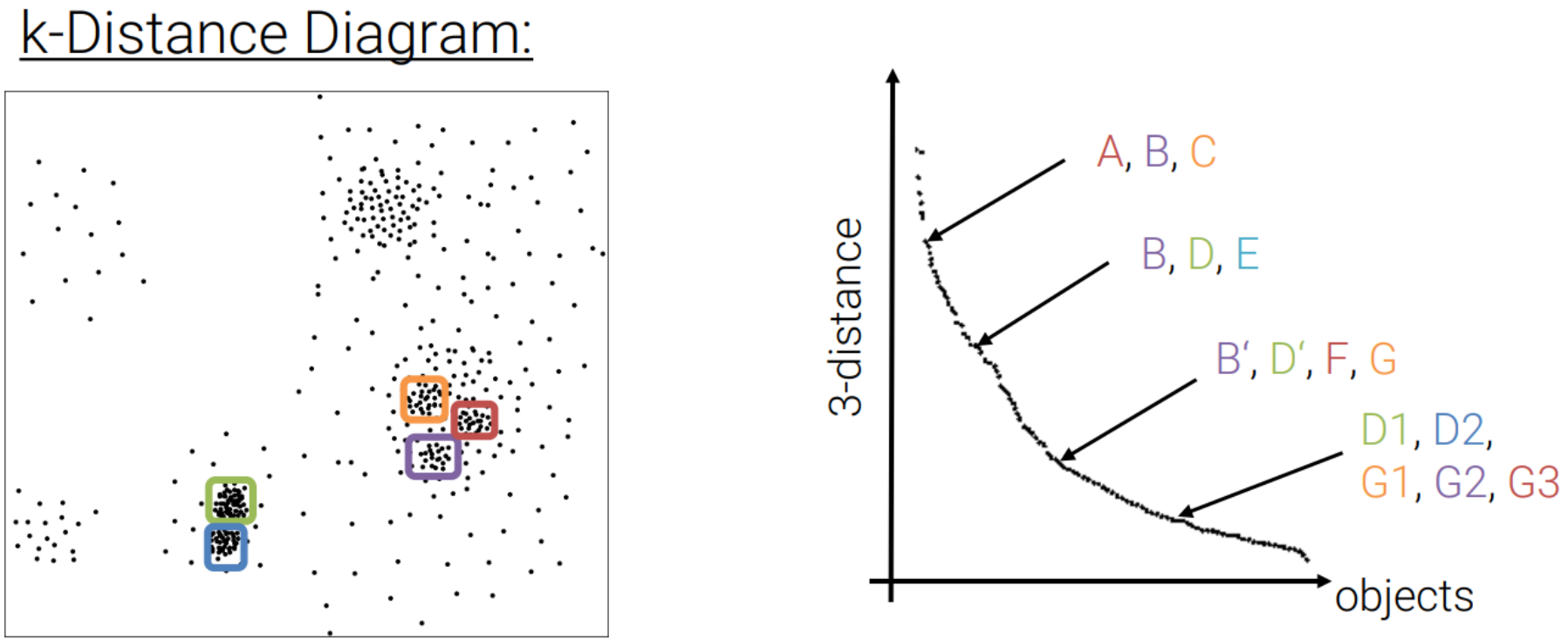


Figure: Behrisch (2021)

Disadvantages

- input parameters may be difficult to determine
- sensitive to input parameter settings
- struggles to cluster datasets with varying densities

Advantages

- can deal with higher dimensions and arbitrary shapes
- number of clusters is determined automatically
- noise / outlier detection
- complexity $O(n^2)$, with spatial index structures: $O(n \log n)$

Outline

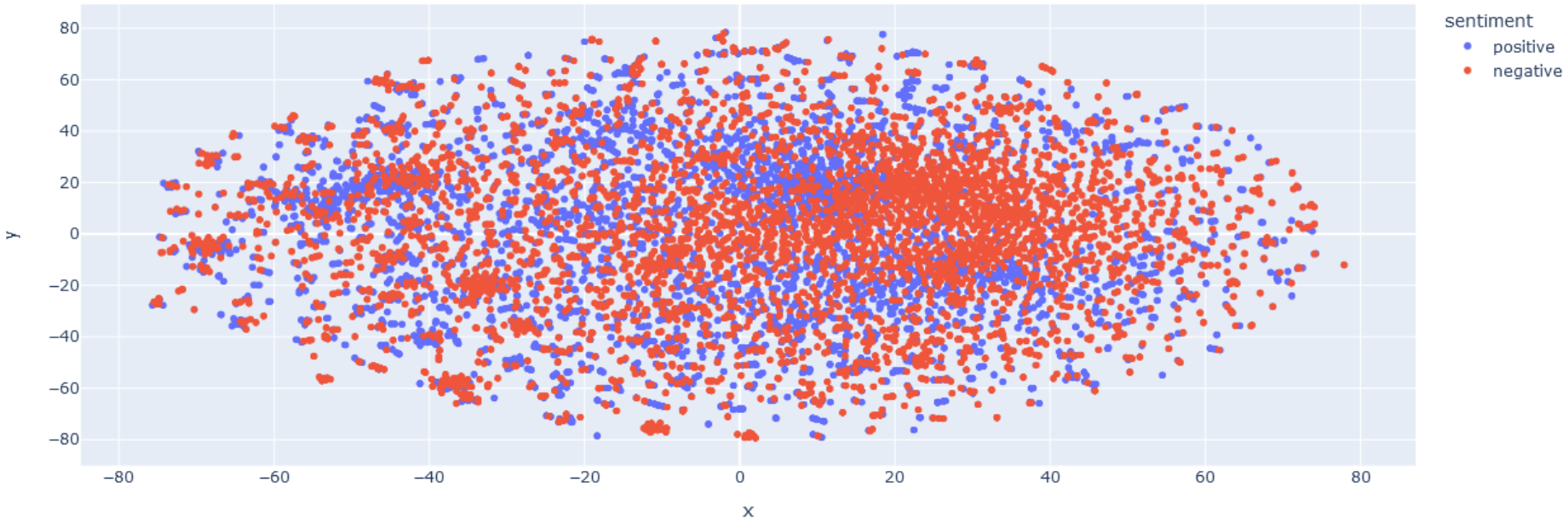
Setup

- Feature Choice: How do we represent the textual data?
- "Old-school" approach: TF-IDF
- Embeddings:
  - Word Embeddings:
    word2vec (Mikolov et al., 2013),
    - `word2vec-google-news-300`

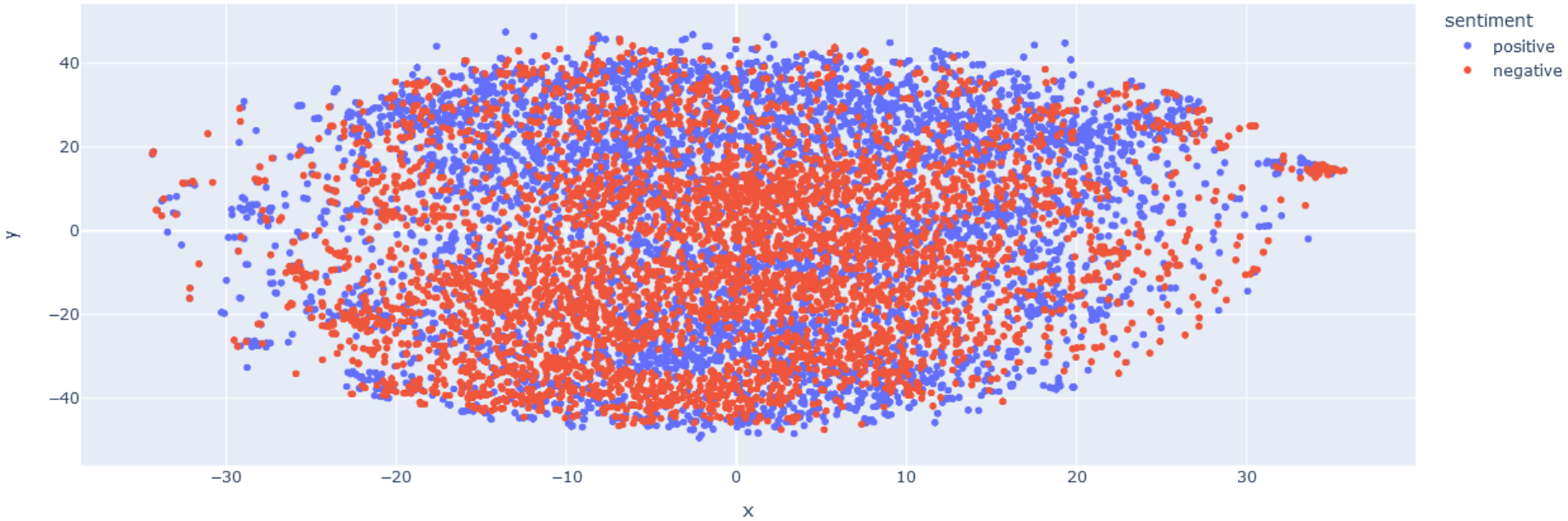
    GloVe (Pennington et al., 2014)
    - `glove-wiki-gigaword-100`
  - Document Embeddings:
    SBERT (Reimers and Gurevych, 2019): all-MiniLM-L6-v2,
    OpenAI: text-embedding-3-small

Lets visualize the embeddings!

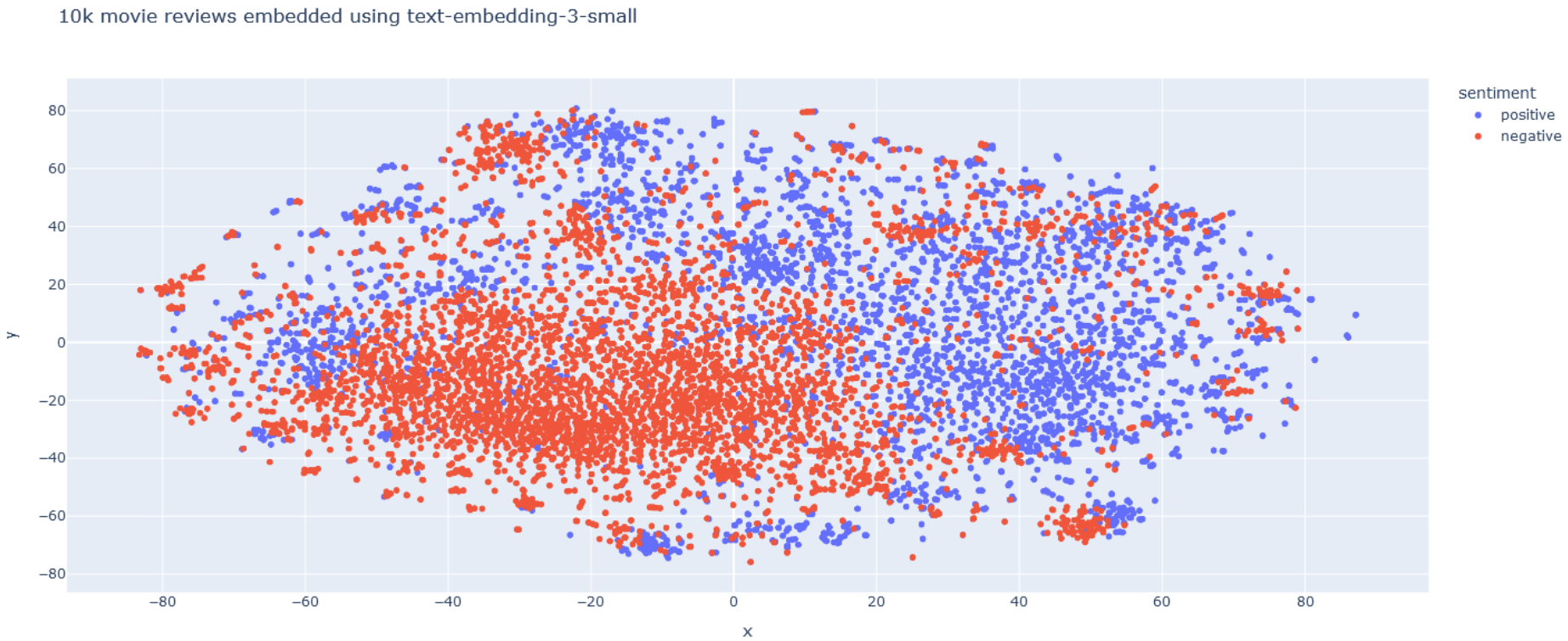
Movie review Representation: TF-IDF



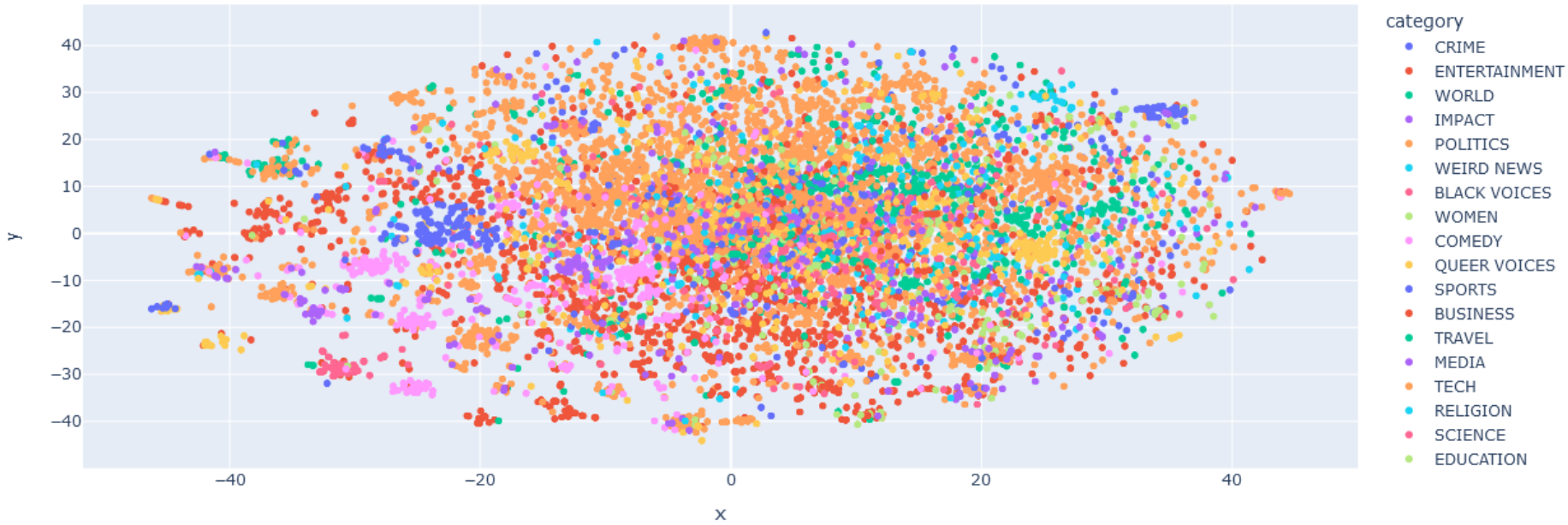
Movie review Embeddings: word2vec



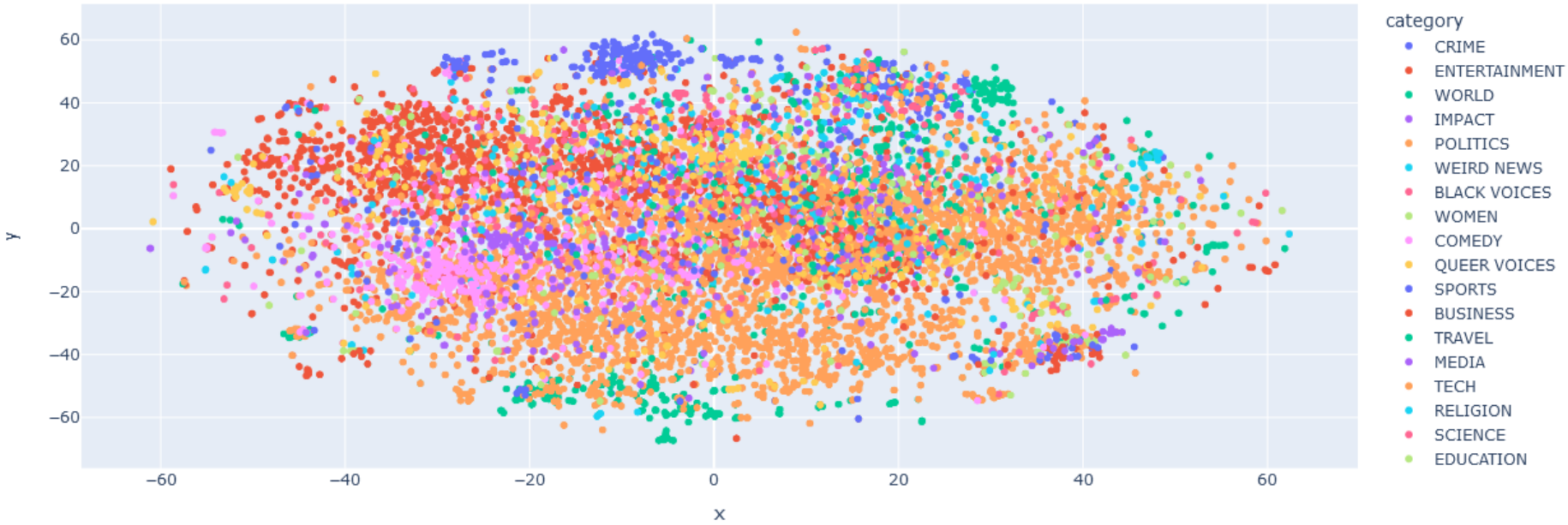
Movie review Embeddings: OpenAI



News headlines representation: TF-IDF

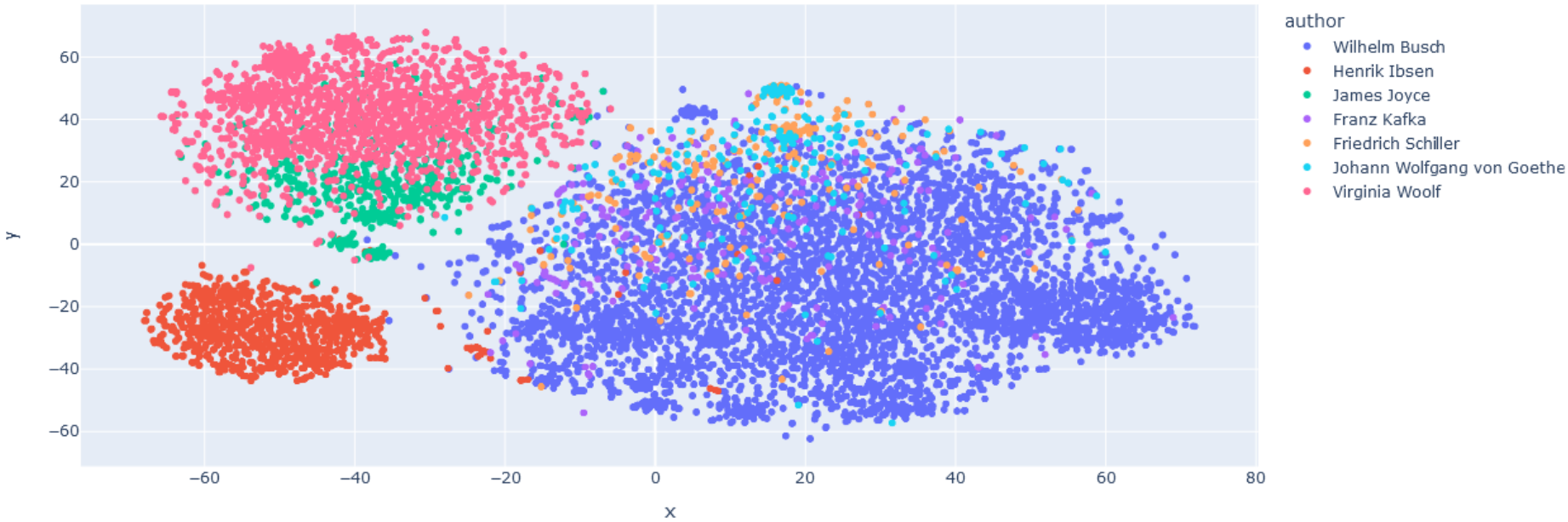


News headlines Embeddings: word2vec

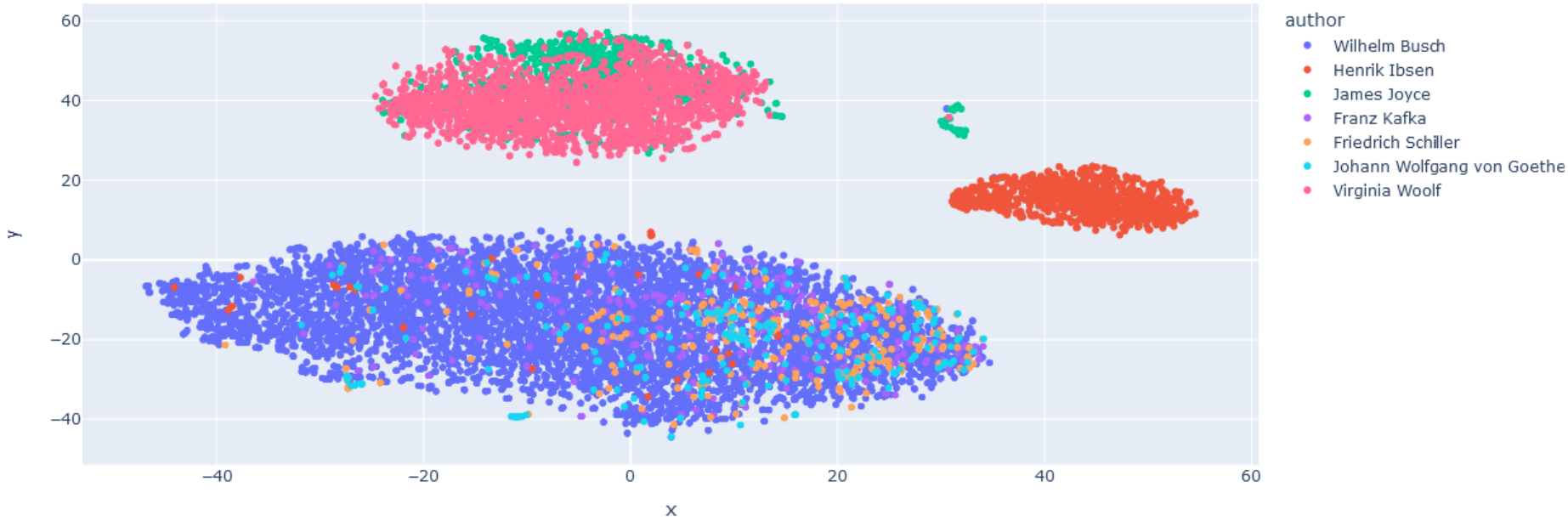


News headlines Embeddings: OpenAI

Letters Representation: TF-IDF



Letters Embeddings: word2vec



Letters Embeddings: OpenAI

Parameter Search

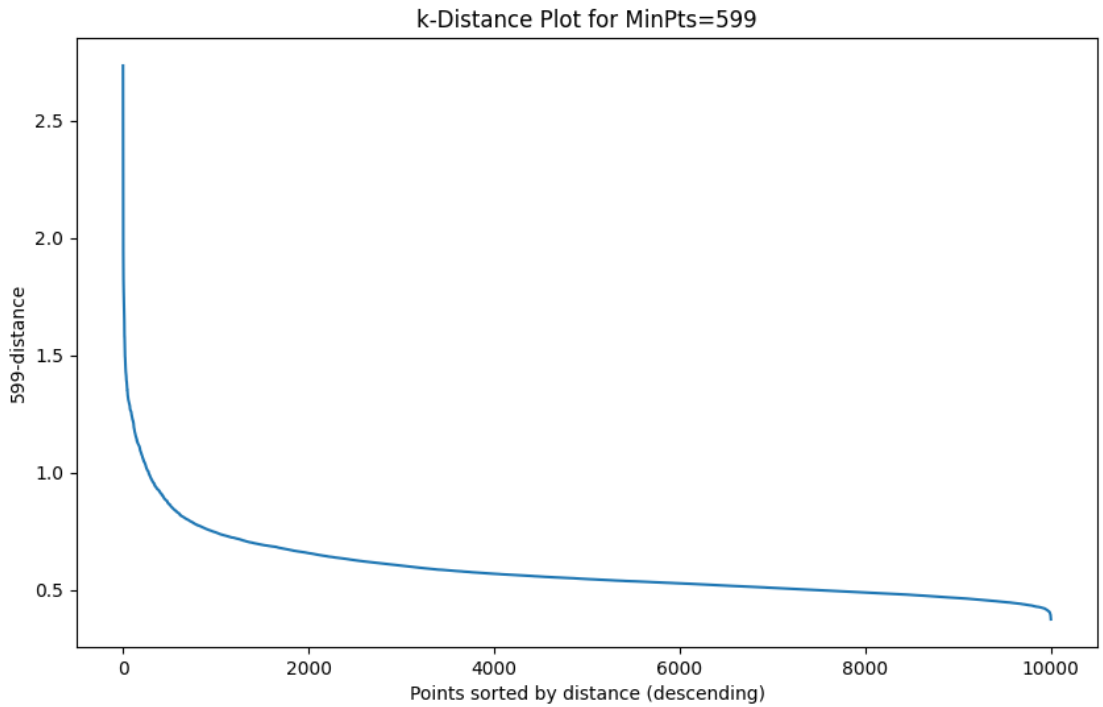


Figure: k-distance graph for word2vec letters w. 599 *MinPts*

- k-distance graph
- additional grid-search based on optimal ε

Outline

Evaluation Metrics

- Internal Metrics
  - Silhouette Score
- External Metrics
  - V-Measure
  - Adjusted Rand Score
  - Purity

Silhouette Score Calculation

Cohesion ($a(i)$): Average distance between i and other points in the same cluster:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq i} d(i, j) \tag{3}$$

Separation ($b(i)$): Minimum average distance between i and points in other clusters:

$$b(i) = \min_{C_k \neq C_i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{4}$$

Silhouette Score for i:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{5}$$

Overall Silhouette Score:

$$S = \frac{1}{N} \sum_{i=1}^{N} s(i) \tag{6}$$

V-Measure Calculation

Homogeneity (h): Clusters contain points from a single true class:

$$h = 1 - \frac{H(C|K)}{H(C)} \tag{7}$$

Completeness (c): Points from a true class belong to the same cluster:

$$c = 1 - \frac{H(K|C)}{H(K)} \tag{8}$$

V-Measure (V): Harmonic mean of h and c:

$$V = \frac{(1+\beta) \cdot h \cdot c}{\beta \cdot h + c} \tag{9}$$

Adjusted Rand Score Calculation

Rand Index (RI): Measures agreement between clustering and ground truth

$$\text{RI} = \frac{\text{Number of agreements}}{\text{Total pairs}} \tag{10}$$

Adjusted Rand Score (ARS):

$$\text{ARS} = \frac{\text{RI} - \text{Expected RI}}{\text{Max RI} - \text{Expected RI}} \tag{11}$$

Formula:

$$\text{ARS} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \tag{12}$$

Purity Calculation

Purity measures the extent to which each cluster contains points from a single true class:

$$\text{Purity} = \frac{1}{N} \sum_{m \in M} \max_{d \in D} |m \cap d| \tag{13}$$

Evaluation: News categories

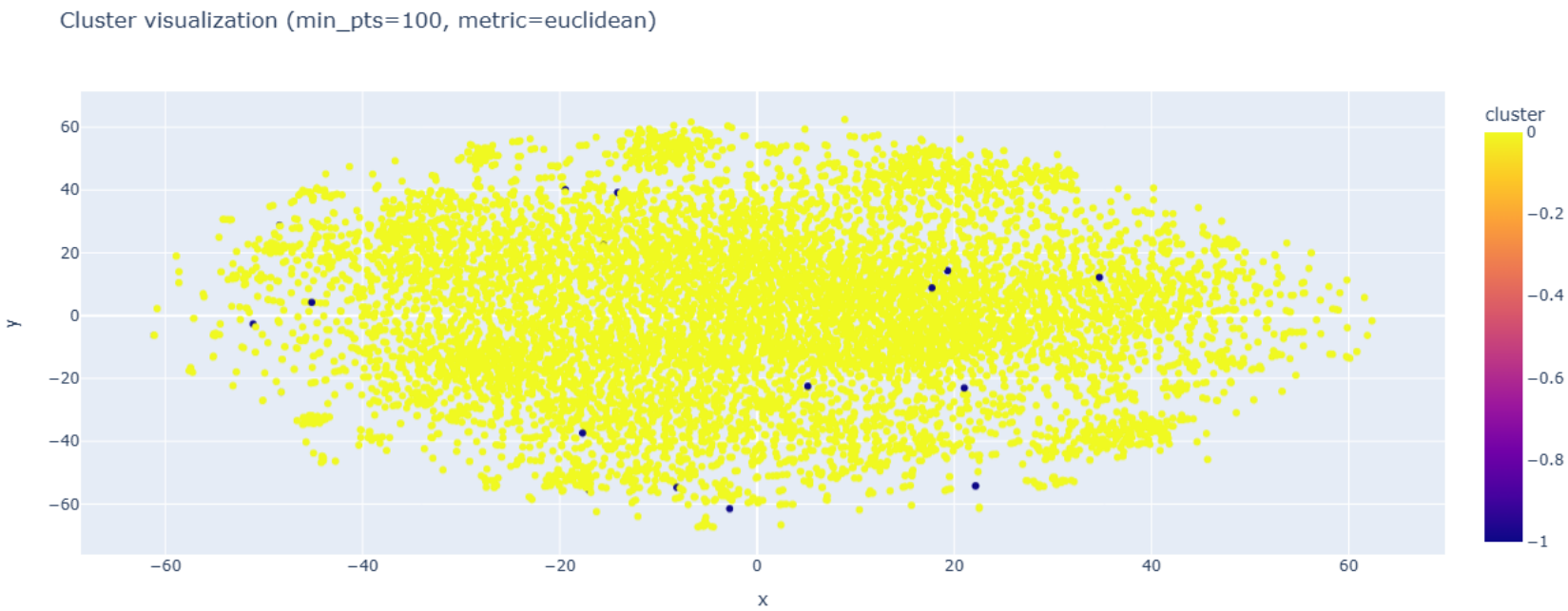
	Silhouette Score	V-Measure	Adjusted Rand Score	Purity
text-embedding-3-small	-1.0000	0.0000	0.0000	0.3658
all-MiniLM-L6-v2	-1.0000	0.0000	0.0000	0.3658
glove-wiki-gigaword-100	-1.0000	0.0000	0.0000	0.3658
word2vec-google-news-300	**0.4147**	**0.0011**	0.0000	0.3650
tf-idf	-1.0000	0.0000	0.0000	0.3657

Table: News headlines DBSCAN clustering evaluation

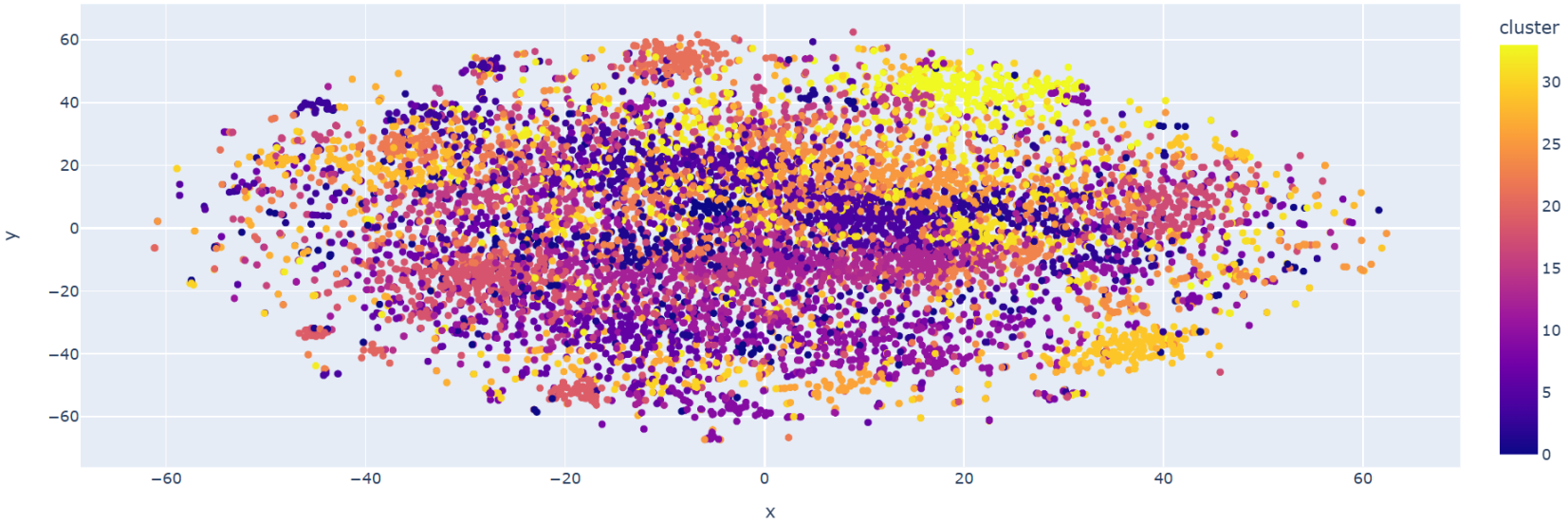
	Silhouette Score	V-Measure	Adjusted Rand Score	Purity
text-embedding-3-small	0.0190	0.1093	**0.3523**	**0.6430**
all-MiniLM-L6-v2	0.0190	0.0806	0.3017	0.5986
glove-wiki-gigaword-100	**0.7216**	0.0484	0.0107	0.3881
word2vec-google-news-300	0.0006	**0.1606**	0.0421	0.4849
tf-idf	0.0191	0.1449	-0.0090	0.4462

Table: News headlines k-means clustering evaluation

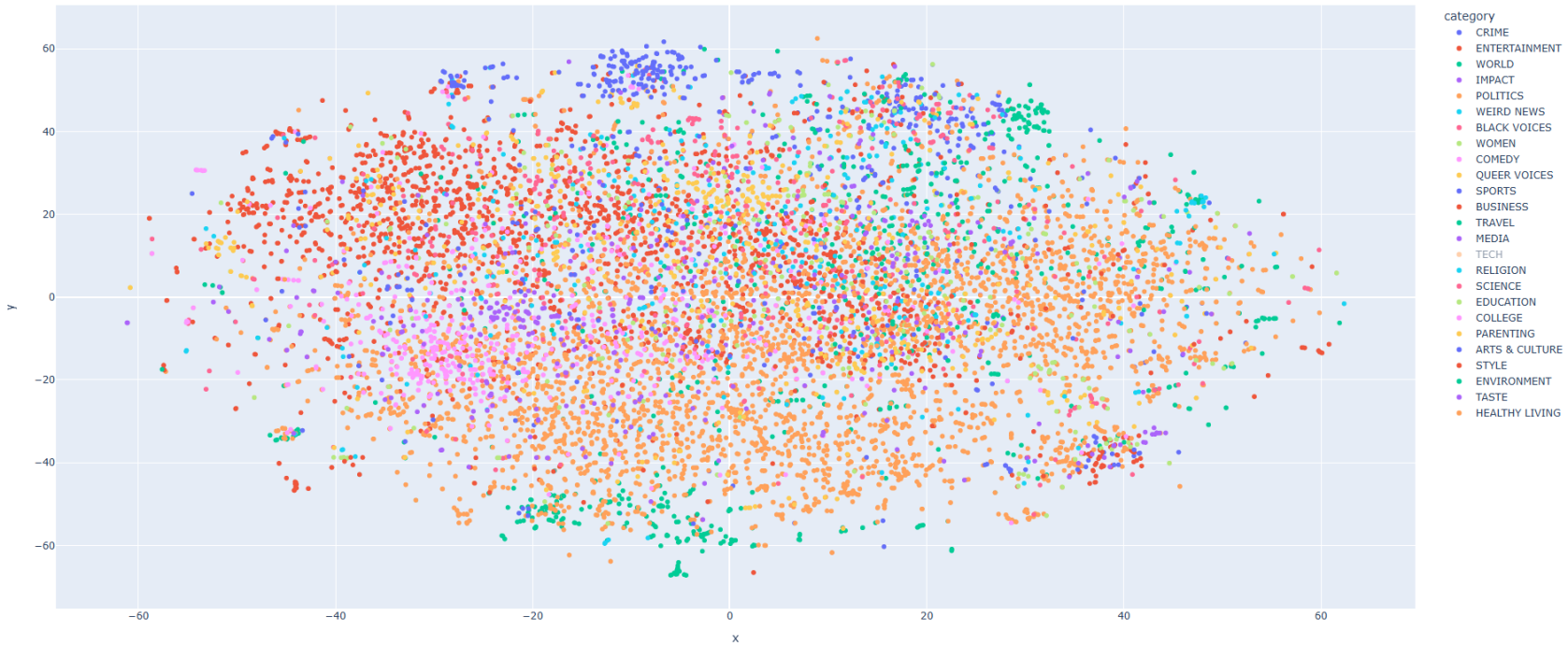
DBSCAN clustering for news with Word2Vec



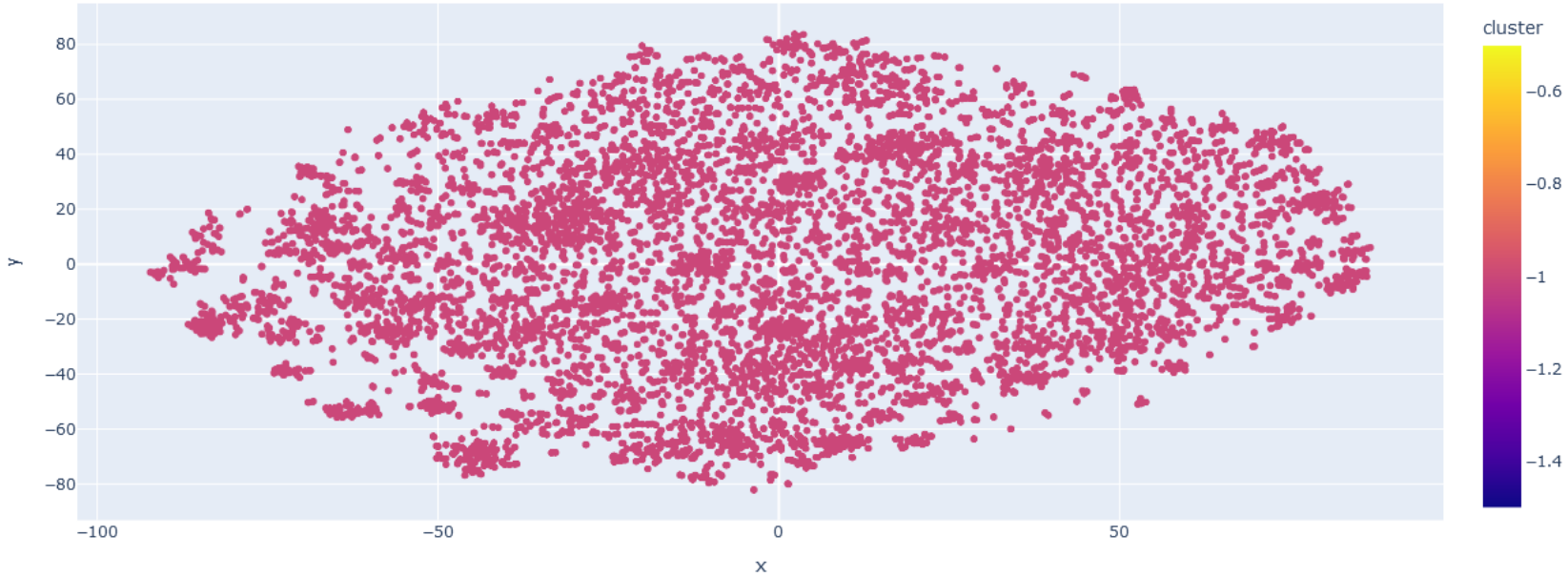
k-means clustering for news with Word2Vec



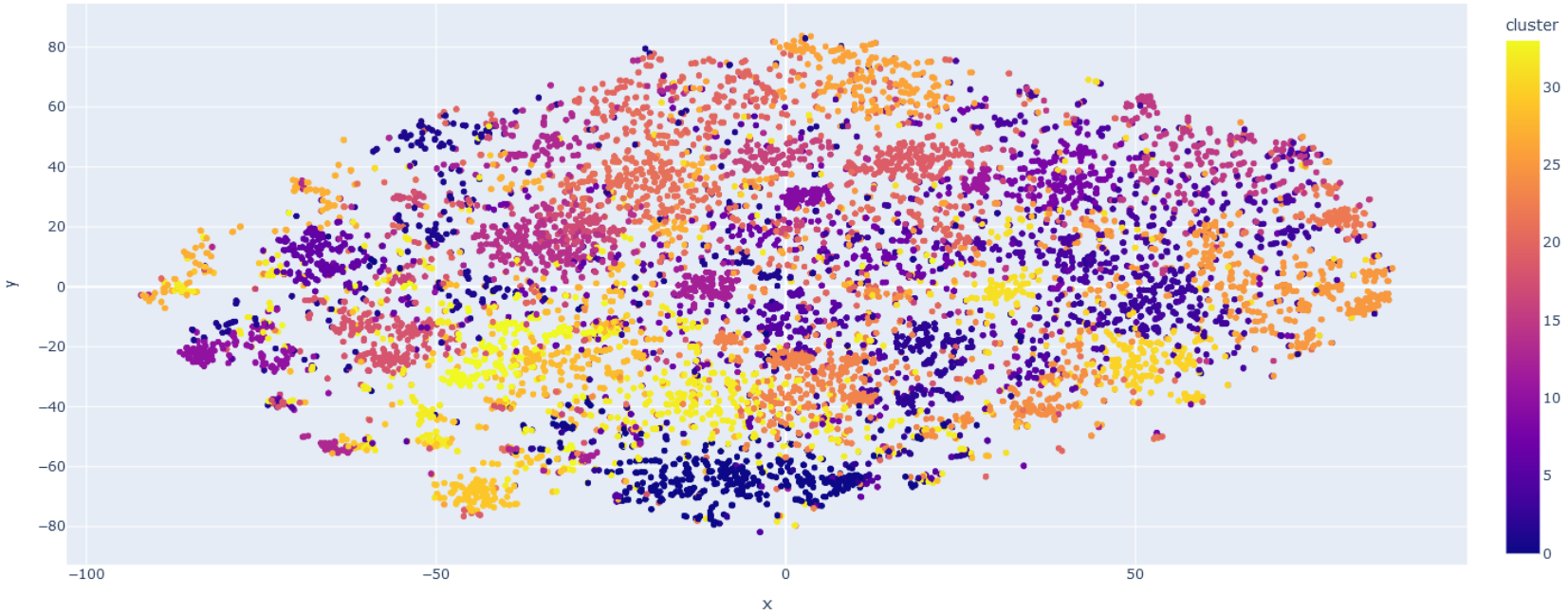
Embeddings for news with Word2Vec



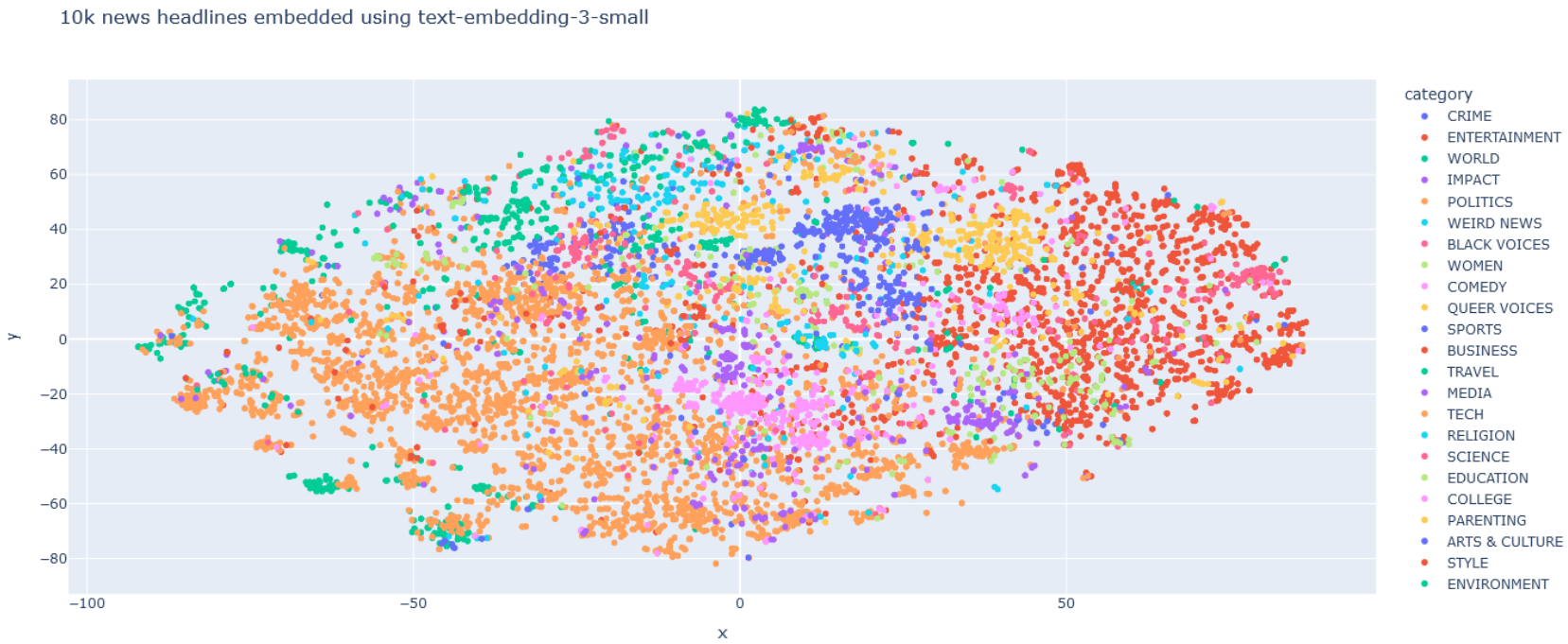
DBSCAN clustering for news with OpenAI



k-means clustering for news with OpenAI



Embeddings for news with OpenAI



Evaluation: Letters

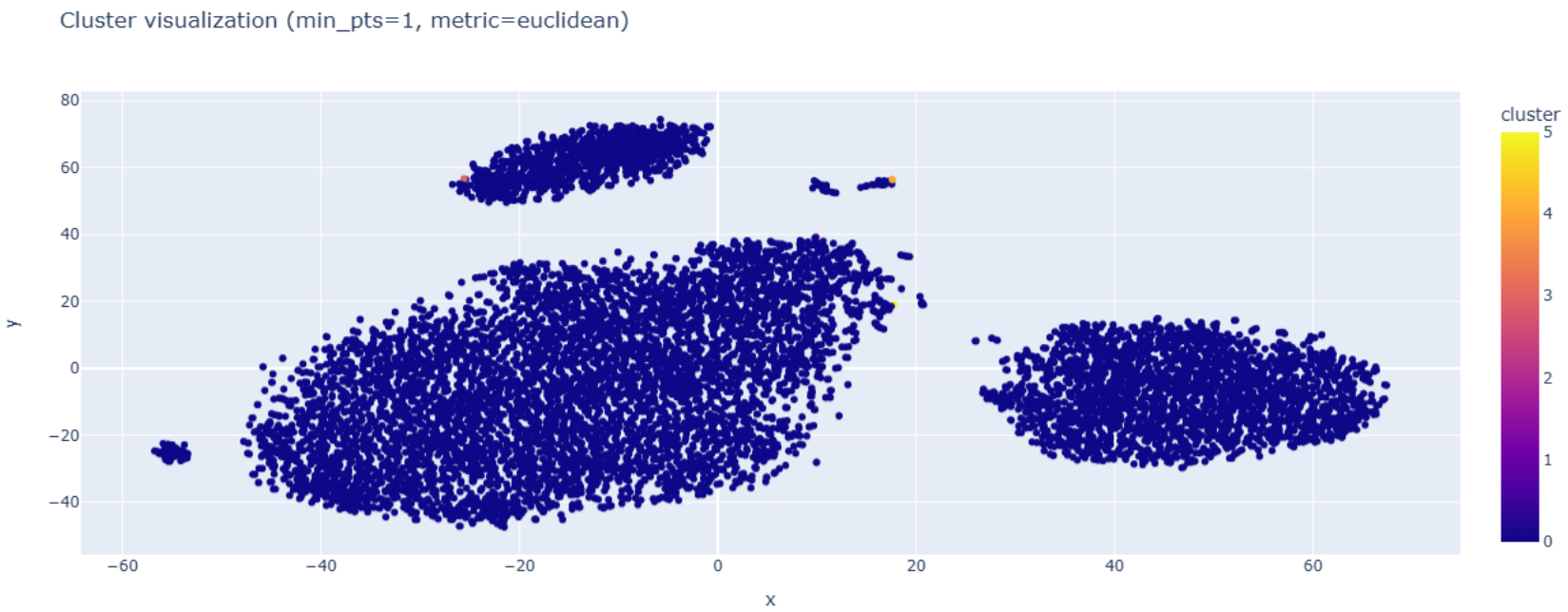
	Silhouette Score	V-Measure	Adjusted Rand Score	Purity
text-embedding-3-small	0.0746	**0.4927**	**0.4094**	**0.7006**
all-MiniLM-L6-v2	0.0568	0.2593	0.3100	0.6170
glove-wiki-gigaword-100	**0.4053**	0.0013	0.0003	0.5641
word2vec-google-news-300	0.3930	0.0014	0.0003	0.5629
tf-idf	-1.0000	0.0000	0.0000	0.5637

Table: Letters DBSCAN clustering evaluation

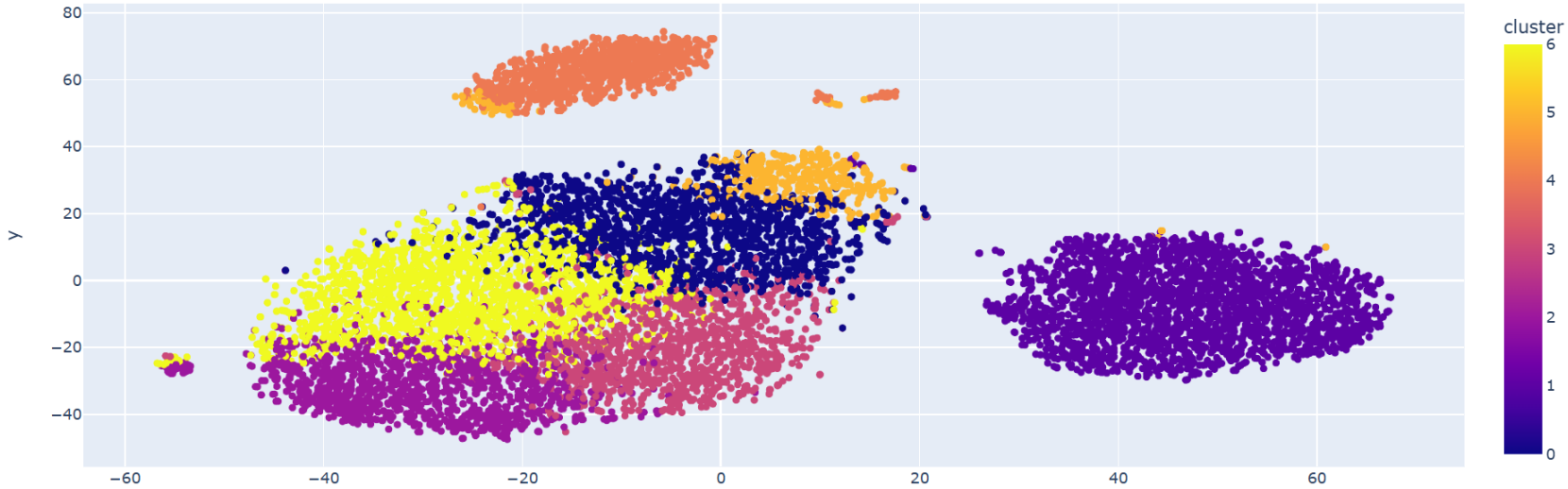
	Silhouette Score	V-Measure	Adjusted Rand Score	Purity
text-embedding-3-small	0.1029	**0.7503**	**0.7598**	0.8499
all-MiniLM-L6-v2	0.0737	0.7372	0.7394	0.8487
glove-wiki-gigaword-100	0.2187	0.5096	0.2848	0.8402
word2vec-google-news-300	**0.2311**	0.5675	0.3562	**0.8535**
tf-idf	0.0120	0.5740	0.3168	0.8502

Table: Letters k-means clustering evaluation

DBSCAN clustering for letters with GloVe

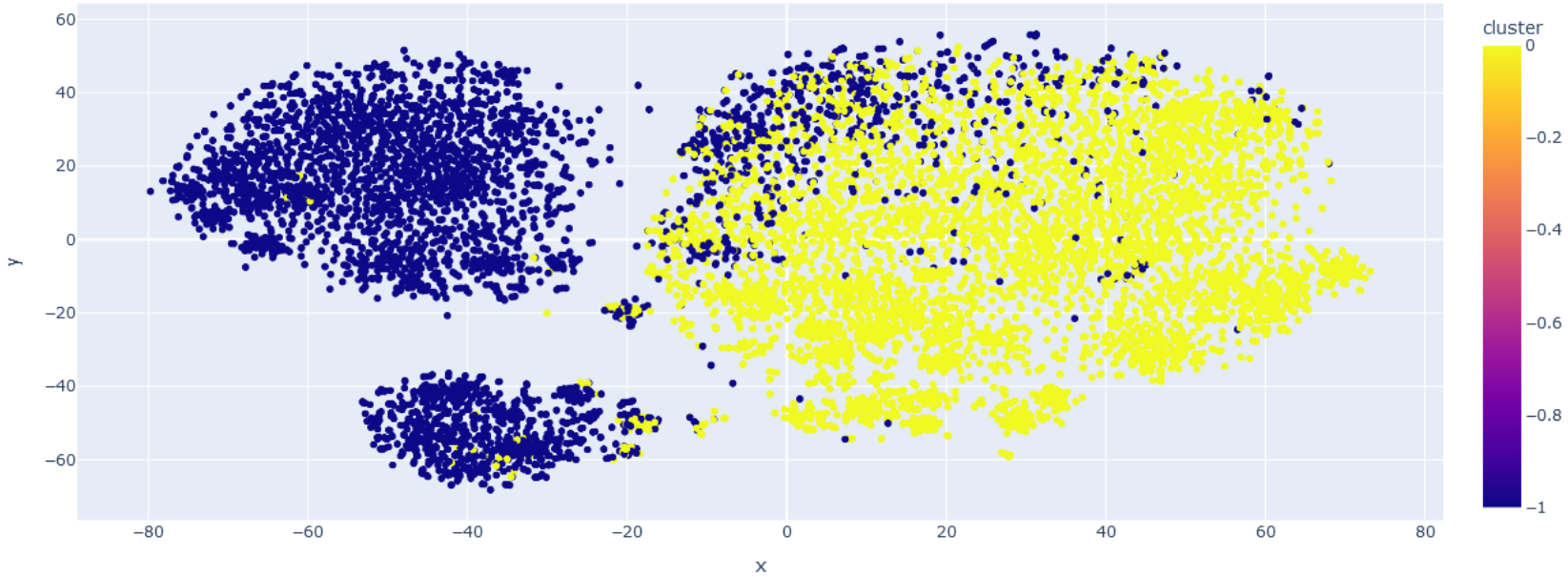


k-means clustering for letters with GloVE

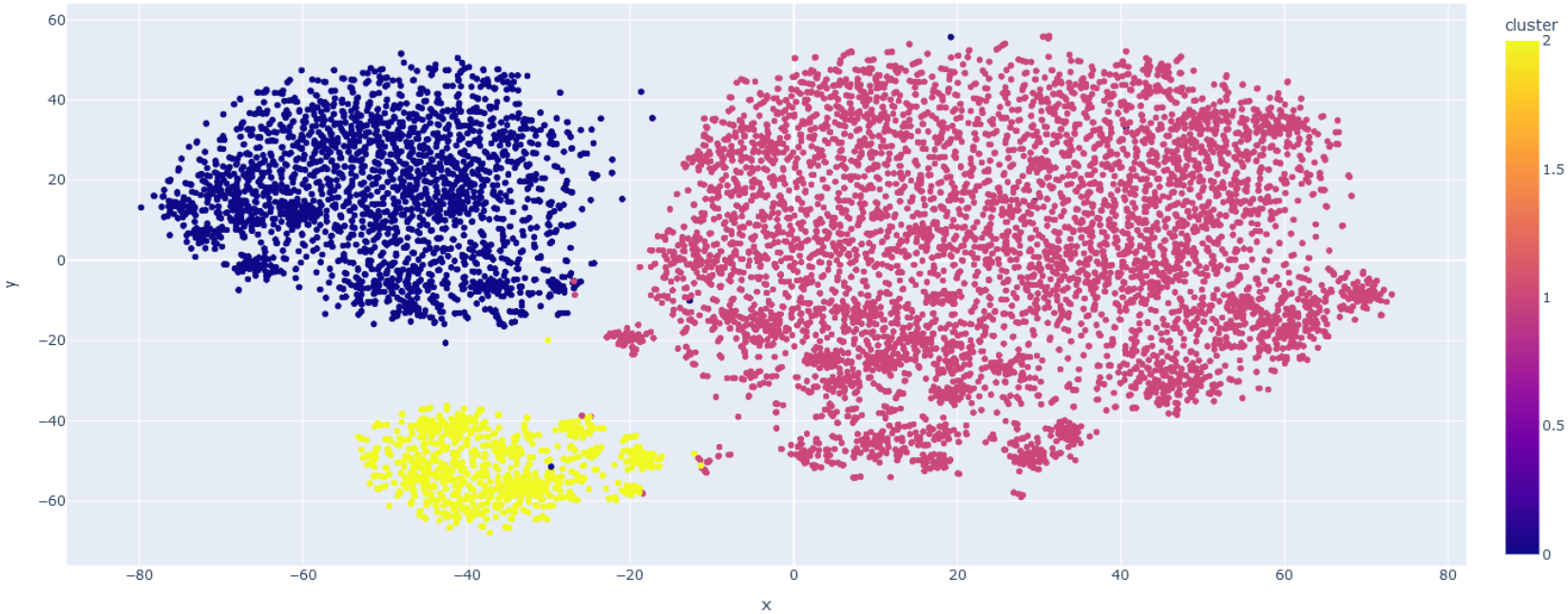


Embeddings for letters with GloVe

DBSCAN clustering for letters with OpenAI



k-means clustering for letters with OpenAI



Embeddings for letters with OpenAI

Evaluation: Movie reviews

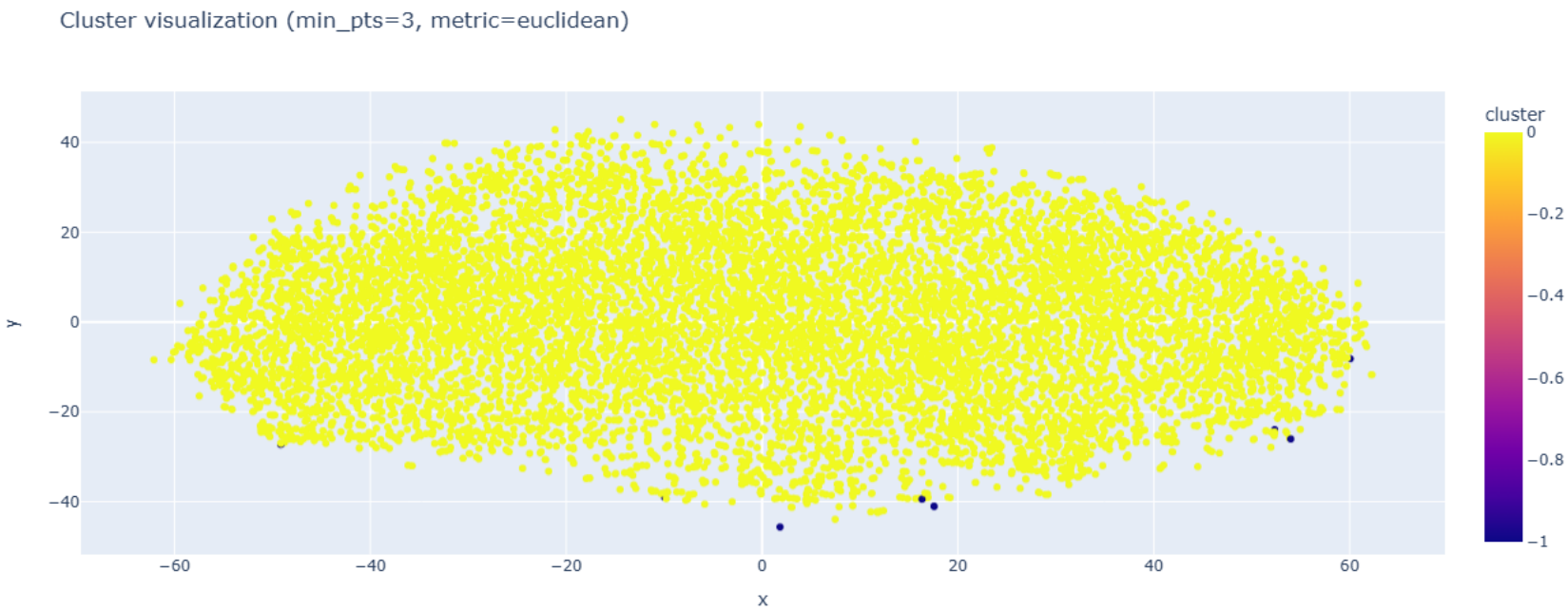
	Silhouette Score	V-Measure	Adjusted Rand Score	Purity
text-embedding-3-small	0.0725	**0.0158**	**0.0556**	**0.5629**
all-MiniLM-L6-v2	0.0617	0.0059	0.0120	0.5387
glove-wiki-gigaword-100	**0.9865**	0.0002	0.0000	0.5030
word2vec-google-news-300	0.5296	0.0000	0.0000	0.5020
tf-idf	-1.0000	0.0000	0.0000	0.5030

Table: Movie reviews DBSCAN clustering evaluation

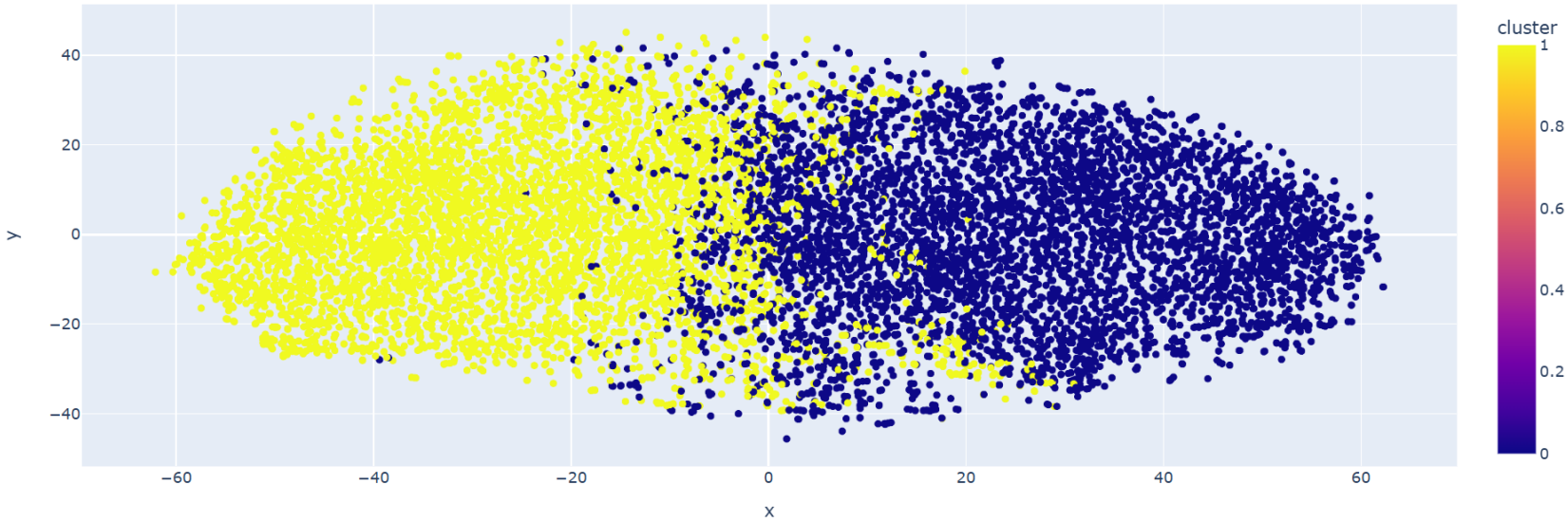
	Silhouette Score	V-Measure	Adjusted Rand Score	Purity
text-embedding-3-small	0.0375	**0.5904**	**0.4914**	**0.8842**
all-MiniLM-L6-v2	0.0294	0.0117	0.0087	0.5542
glove-wiki-gigaword-100	**0.1254**	0.0165	0.0227	0.5755
word2vec-google-news-300	0.0977	0.0230	0.0316	0.5890
tf-idf	0.0030	0.0165	0.0183	0.5678

Table: Movie reviews k-means clustering evaluation

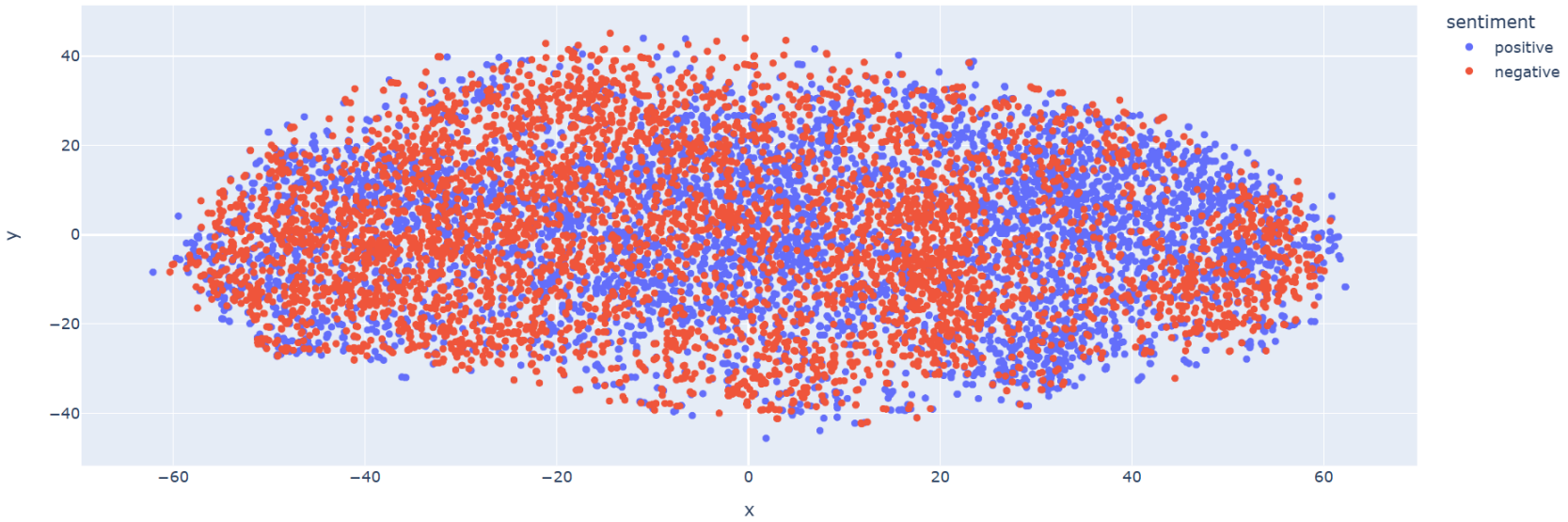
DBSCAN clustering for reviews with GloVe



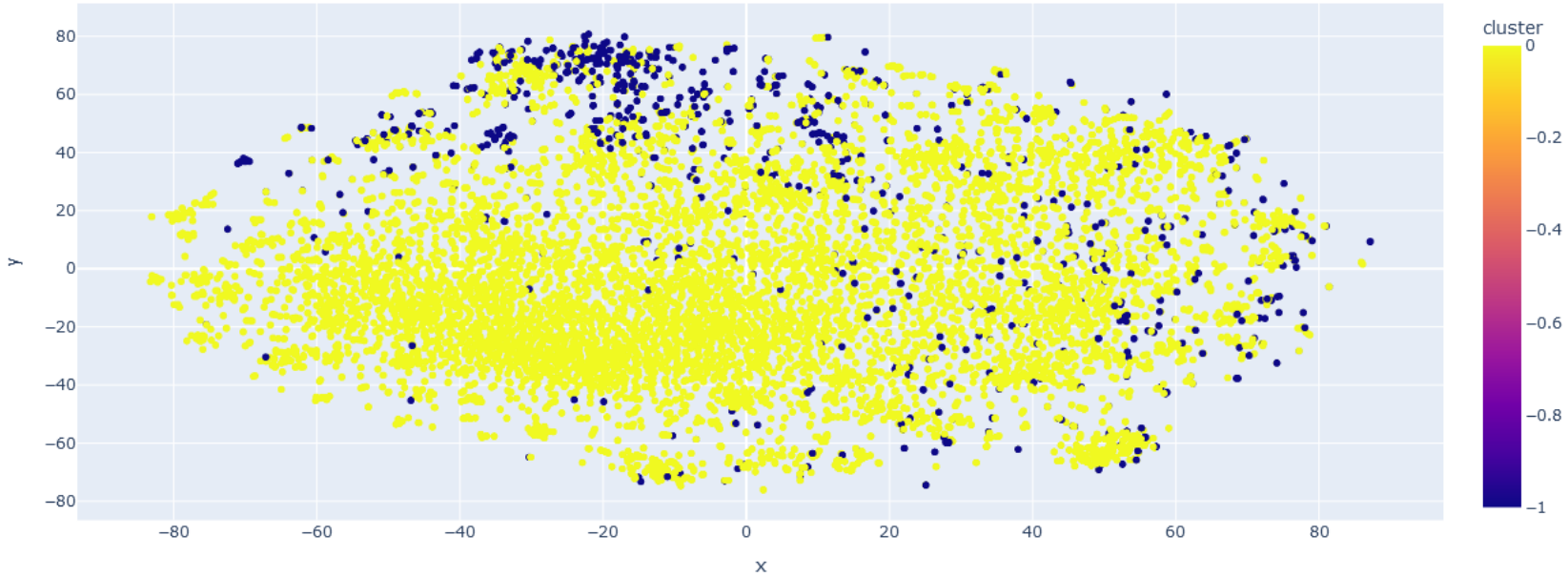
k-means clustering for reviews with GloVe



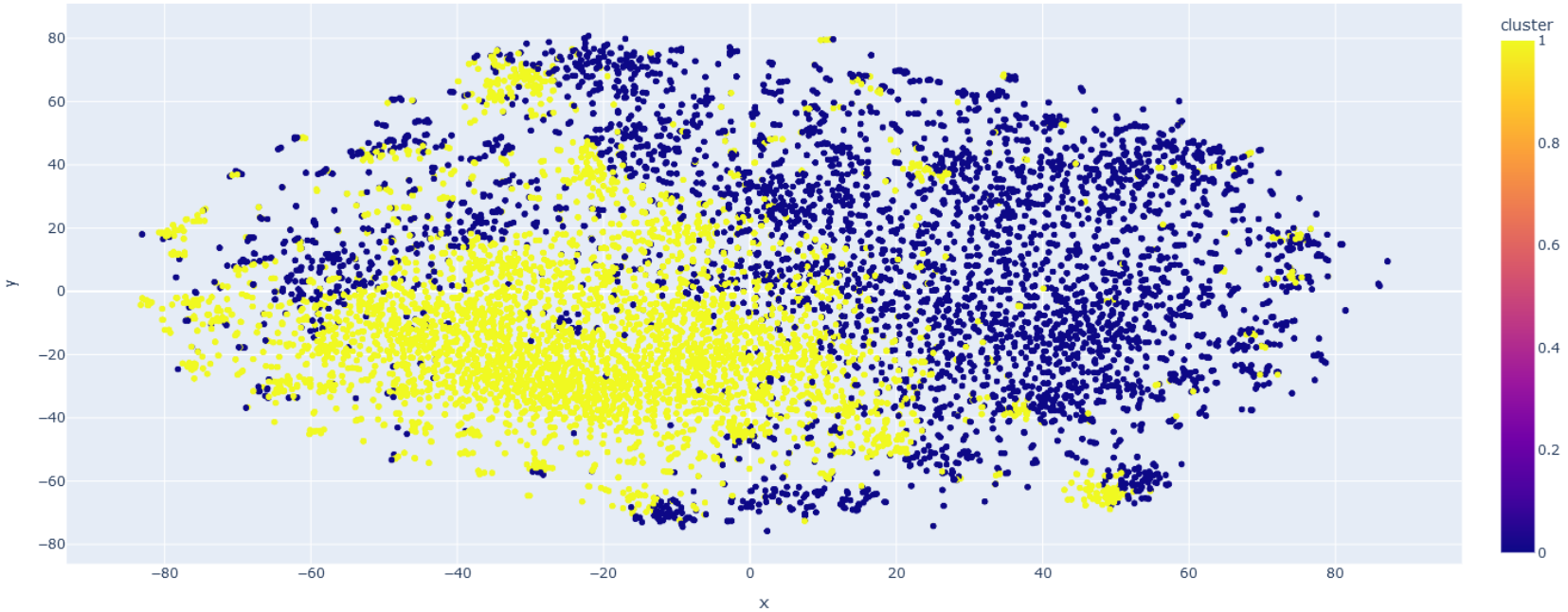
Embeddings for reviews with GloVe



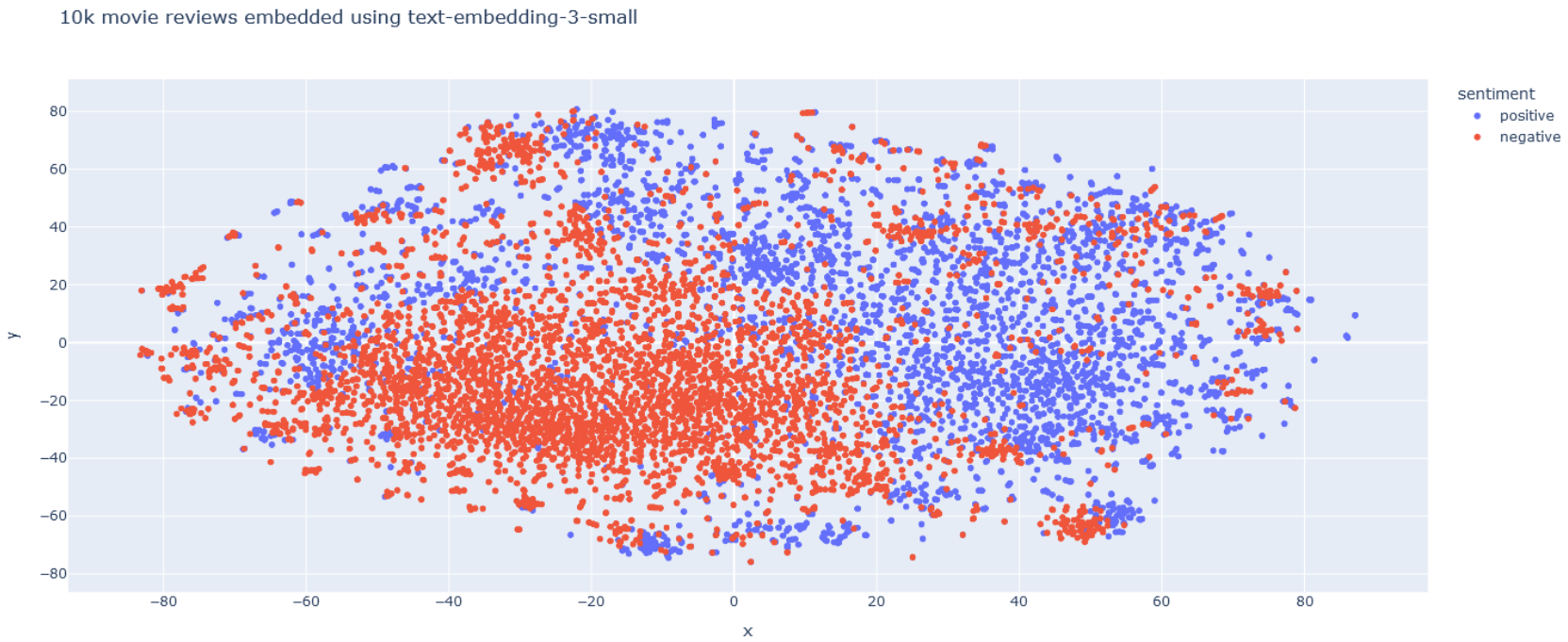
DBSCAN clustering for reviews with OpenAI



k-means clustering for reviews with OpenAI



Embeddings for reviews with OpenAI



Outline

Discussion: General

- There are many different ways to cluster data:
  - Use different data representations, different clustering algorithms, different hyperparameters ...
  - Finding the right setup can be challenging

Discussion: General

- There are many different ways to cluster data:
  - Use different data representations, different clustering algorithms, different hyperparameters ...
  - Finding the right setup can be challenging
- Some clustering algorithms only work well in specific scenarios:

Discussion: General

- There are many different ways to cluster data:
  - Use different data representations, different clustering algorithms, different hyperparameters ...
  - Finding the right setup can be challenging
- Some clustering algorithms only work well in specific scenarios:
  - Know your data!

Discussion: General

- There are many different ways to cluster data:
  - Use different data representations, different clustering algorithms, different hyperparameters ...
  - Finding the right setup can be challenging
- Some clustering algorithms only work well in specific scenarios:
  - Know your data!
  - But: How do we know our data before clustering?

Discussion: General

- There are many different ways to cluster data:
  - Use different data representations, different clustering algorithms, different hyperparameters ...
  - Finding the right setup can be challenging
- Some clustering algorithms only work well in specific scenarios:
  - Know your data!
  - But: How do we know our data before clustering?
- You cannot "force" clusters on any dataset:
  If the data does not contain distinct clusters, no algorithm will yield meaningful results

Discussion: DBSCAN

- For textual data, DBSCAN did not perform well.

Discussion: DBSCAN

- For textual data, DBSCAN did not perform well.
  - High-dimensional embeddings are typically not uniformly distributed in the vector space, meaning they show little variance and are therefore challenging to cluster
    $\rightarrow$ **Anisotropy problem**

Discussion: DBSCAN

- For textual data, DBSCAN did not perform well.
  - High-dimensional embeddings are typically not uniformly distributed in the vector space, meaning they show little variance and are therefore challenging to cluster
    $\rightarrow$ **Anisotropy problem**
  - DBSCAN works well at detecting outliers, but the given data did not contain outliers

Discussion: DBSCAN

- For textual data, DBSCAN did not perform well.
  - High-dimensional embeddings are typically not uniformly distributed in the vector space, meaning they show little variance and are therefore challenging to cluster
    $\rightarrow$ **Anisotropy problem**
  - DBSCAN works well at detecting outliers, but the given data did not contain outliers
- DBSCAN is very sensitive to hyperparameters:
  - There are ways to approximate good parameters,
    but the "right" choice still requires experimentation or may not exist in the first place

DBSCAN actually working!!

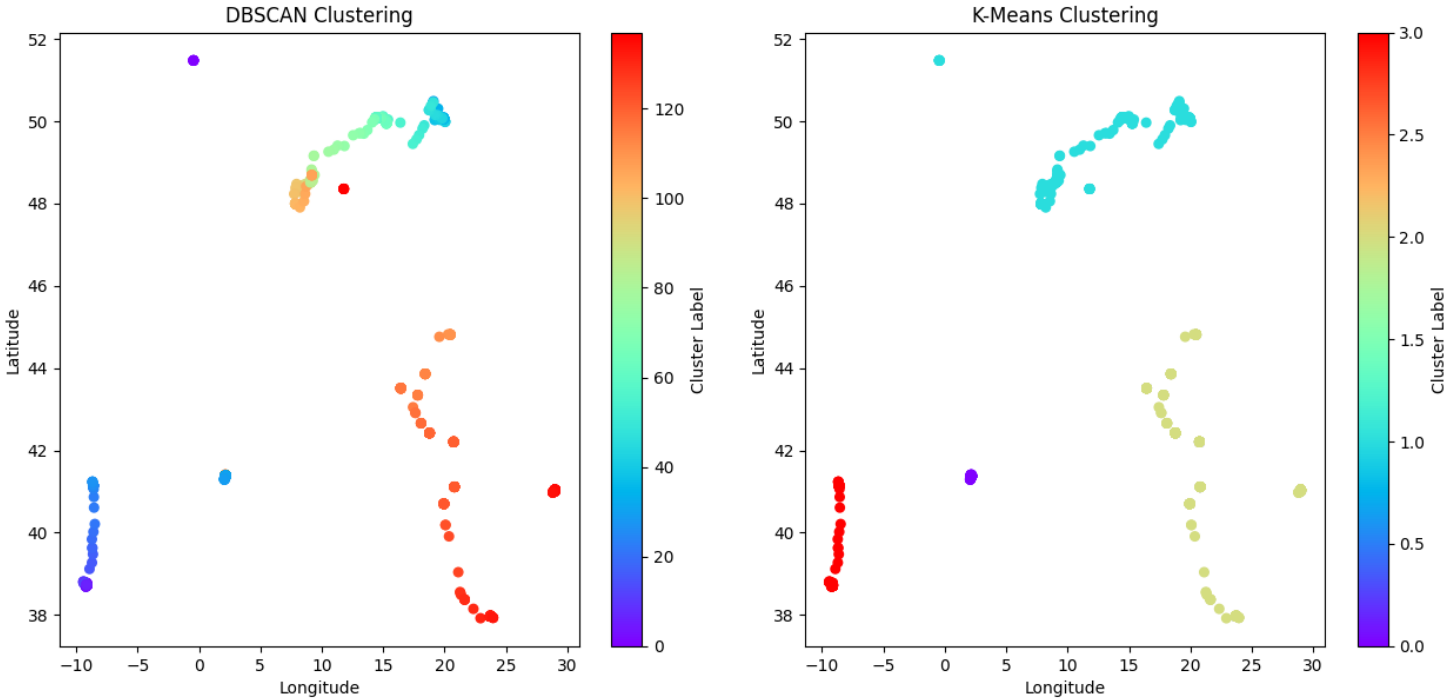


Figure: Clustering based on geographic data

Thank you so much for your attention!

References I

Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60.

Michael Behrisch. 2021. Data analytics: Clustering - density-based methods. Lecture at Utrecht University. Accessed: 2024-12-15.

Derya Birant and Alp Kut. 2007. St-dbscan: An algorithm for clustering spatial–temporal data. *Data & knowledge engineering*, 60(1):208–221.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer.

Martin Ester. 2018. *Density-Based Clustering*, pages 1053–1058. Springer New York, New York, NY.

References II

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

References III

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. 1998. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery*, 2:169–194.

Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Josh Starmer. 2022. Clustering with dbscan, clearly explained!!! YouTube video. Accessed: 2024-12-15.

Additional Slides

Clustering Algorithms Based on DBSCAN

- **OPTICS (Ordering Points to Identify Clustering Structure):** Extends DBSCAN by handling clusters with varying densities (Ankerst et al., 1999)
- **HDBSCAN (Hierarchical DBSCAN):** Improves DBSCAN by building a hierarchical cluster tree and selecting clusters based on stability (Campello et al., 2013)
- **GDBSCAN (Generalized DBSCAN):** Adapts DBSCAN to various data types by generalizing the concept of density (Sander et al., 1998)
- **ST-DBSCAN (Spatial-Temporal DBSCAN):** Incorporates spatial and temporal dimensions for clustering spatiotemporal data (Birant and Kut, 2007)