# Topological Structure in GitHub Repository Embeddings Using Mapper

Ivo Hrib
ivo.hrib@gmail.com
Jožef Stefan Institute
Ljubljana, Slovenia

Patrik Zajec
patrik.zajec@ijs.si
Jožef Stefan Institute
Ljubljana, Slovenia

## Abstract

We present a preliminary framework for the topological analysis of GitHub repository embeddings using the Mapper algorithm. Applied to 10,000 repositories embedded in 768-dimensional space, our approach currently provides visual representations of Mapper graphs, offering a first view into potential topological structures such as branching patterns and cycles. While these initial results are exploratory, they establish a foundation for rigorous statistical testing of topological features. Future work will incorporate persistent homology–based significance testing to distinguish genuine structural patterns from noise, with the ultimate goal of interpreting these features in terms of repository characteristics.

## Keywords

topological data analysis, Mapper, GitHub, embeddings, significance testing, software repositories, persistent homology

## 1 Introduction

We present a preliminary framework for the topological analysis of GitHub repository embeddings using the Mapper algorithm. Applied to 10,000 repositories embedded in 768-dimensional space, our approach provides visual representations of Mapper graphs that reveal branching structures and cycles as potential organizational patterns in the data. These results are exploratory and serve as a foundation for future work, where statistical significance testing will be applied to rigorously validate which features represent genuine topological structure rather than noise. Our framework thus establishes an initial step toward understanding the topology of repository embeddings and motivates further methodological development.

### 1.1 Research Questions

This work addresses the following specific question:

(1) Do GitHub repository embeddings contain significant topological structures beyond simple clustering?

### 1.2 Contributions

Our main contributions are:

- A preliminary framework for constructing and visualizing Mapper graphs of GitHub repository embeddings.
- A systematic comparison of Mapper graphs across multiple parameter settings, highlighting sensitivity and recurring structural patterns.
- A discussion of how these preliminary results can guide future work, in particular the application of statistical testing methods to validate topological features and their interpretation in terms of repository characteristics.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
*Information Society 2025, Ljubljana, Slovenia*
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.70314/is.2025.sikdd.27

## 2 Background and Related Work

### 2.1 The Mapper Algorithm

The Mapper algorithm [6] constructs a graph representation of a topological space by combining a filter function, overlapping covers, and clustering. Given a point cloud $P$ embedded in $\mathbb{R}^d$ and a continuous function $f : P \to \mathbb{R}$ refered to as a filter function, the algorithm:

(1) Constructs a cover $\mathcal{U} = \{U_1, \ldots, U_n\}$ of the range $f(P)$ using overlapping intervals
(2) For each interval $U_i$, computes the preimage $P_{U_i} = f^{-1}(U_i)$
(3) Clusters each preimage into connected components using a clustering algorithm
(4) Creates vertices for each cluster and edges between clusters whose point sets intersect

Common practice uses the first PCA component [4] as the filter and density based clustering methods, such as DBSCAN [2], unless specific domain knowledge is provided. The resulting graph $G = (V, E)$ provides a combinatorial description with mapping $\phi : V \to \mathcal{P}(P)$ associating each vertex with a subset of points.

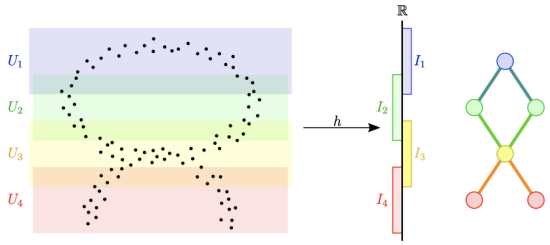



**Figure 1: Visual demonstration of mapper algorithm for a projection filter and a simple pointcloud.**

*2.1.1 Parameter Selection and Sensitivity.* Mapper results are sensitive to three main parameters:

- **Resolution** ($n$): Number of intervals in the cover
- **Overlap** ($p$): Percentage overlap between consecutive intervals
- **Clustering threshold** ($\epsilon$): Distance parameter for the clustering algorithm

Taking this into account, we devised the following methodology for parameter selection. We define a discrete grid of candidate values, for which the mapper graph is reasonably computable, for each of the previously mentioned parameters and the minimum number of points per cluster. For each point in this grid

we applied the Mapper algorithm to the dataset and computed a collection of quality measures.

Three main criteria were used to assess the quality of each Mapper graph:

- **Coverage:** the proportion of data points captured by the nodes of the Mapper graph, measuring how well the graph represents the entire dataset.
- **Modularity:** a measure of the strength of community structure in the resulting graph, reflecting the presence of well-defined clusters or substructures.
- **Stability:** the reproducibility of the graph under sampling noise, estimated by a bootstrap procedure in which multiple resampled datasets were processed and the resulting node assignments compared for consistency.

For each parameter combination, we computed stability, coverage, and modularity. To aggregate these into a single composite score, we used a weighted sum that places the highest emphasis on stability (0.5), followed by coverage (0.3) and modularity (0.2). These weights were chosen to reflect our prioritization of reproducibility and representativeness over community structure.

| $n_{cubes}$ | Overlap | $\varepsilon$ | MinPts | Coverage | Stability | Modularity | Score |
|---|---|---|---|---|---|---|---|
| 12 | 0.70 | 0.7 | 3 | 0.966 | 0.948 | 0.785 | 0.921 |
| 12 | 0.70 | 0.7 | 5 | 0.933 | 0.924 | 0.745 | 0.891 |
| 16 | 0.70 | 0.7 | 3 | 0.952 | 0.872 | 0.791 | 0.880 |
| 10 | 0.70 | 0.7 | 3 | 0.966 | 0.847 | 0.765 | 0.866 |
| 16 | 0.70 | 0.7 | 5 | 0.915 | 0.852 | 0.771 | 0.855 |

**Table 1: Top 5 Mapper parameter settings ranked by score.**

For each parameter layout, we employed the first PCA component as our chosen filter and DBSCAN as our chosen clustering algorithm.

*2.1.2 Adaptive Mapper and Learnable Filter Functions.* Recent advances in Mapper methodology include adaptive approaches in which filter functions are learned from data rather than manually specified. Such approaches could potentially optimize for statistically significant topological features[3]. These methods were, however, not utilized in our case due to computational complexity and remain to be explored in the future.

## 2.2 Related Work in Software Repository Analysis

Several recent studies have explored software repository embeddings and clustering. For example, Rokon et al. introduced *Repo2Vec*, which combines metadata, source code, and structural signals into repository embeddings for similarity search and clustering [5]. Lherondelle et al. proposed an attention-based model that learns repository embeddings from code and metadata to support auto-tagging and recommendation tasks [**lherondelle2022topical**]. Zhang et al. developed *HiGitClass*, a hierarchical classification framework for GitHub repositories using embedding-based methods [8]. Other work has examined clustering repositories with software metrics [**repo_metrics2023**] or analyzing the characteristics of repositories in specific domains such as embedded systems [**polaczek2021embedded**].

While these approaches demonstrate that embeddings and clustering can yield useful insights about software repositories, they focus primarily on supervised tasks (classification, tagging) or flat similarity clustering. In contrast, our work explores the topological features of the high-dimensional space in which repositories reside. By applying the Mapper algorithm to repository embeddings, we intend to characterize how repositories are organized in terms of branching patterns, hubs, and cycles. This perspective emphasizes the geometry and connectivity of the embedding space itself, offering potential insights that complement more conventional similarity- or classification-based analyses of repositories.

# 3 Dataset and Methodology

## 3.1 Dataset Description

The raw dataset comprised approximately 500,000 GitHub repositories, each annotated with a range of metadata fields. These can be grouped into three broad categories:

- **Textual features:** free-form text fields such as description, readme, requirements, and packages, which capture natural-language documentation and dependency declarations.
- **Categorical features:** attributes such as language, topic, and visibility, which provide discrete labels describing repository characteristics.
- **Contextual metadata:** fields such as name, bio, website, company, location, and date of creation, which provide identifying information and organizational context.

*3.1.1 Repository Selection Criteria.* In the interest of computational feasibility, this dataset was then sampled to 10,000 repositories. Repositories were chosen via simple random sampling from the full dataset, as many repositories contained incomplete or inconsistent categorical and contextual metadata; therefore, stratified sampling was not appropriate.

*3.1.2 Embedding Process.* Each sampled repository was converted into a structured dictionary combining the available metadata fields. These dictionaries were embedded using the *nomic-embed-text* model. The model accepts long-context inputs (up to approximately 8,000 tokens), which makes it suitable for processing repository documentation such as README files.

The resulting embeddings are 768-dimensional vectors. Together, the 10,000 sampled repositories form a point cloud in $\mathbb{R}^{768}$. Because the embeddings primarily reflect textual and documentation content (e.g., README and description fields), the analysis in this study centers on topological structure in the documentation space rather than source code semantics. These embeddings serve as the basis for the Mapper-based topological data analysis described in the following sections.

## 3.2 Mapper Implementation

For our purposes we used Kepler-Mapper to get the mapper graphs which scored highest, previously mentioned in **??**.

- Graph 1:
  Resolution = 12, Overlap = 0.7, eps = 0.7, min_samples = 3
- Graph 2:
  Resolution = 12, Overlap = 0.7, eps = 0.7, min_samples = 5
- Graph 3:
  Resolution = 16, Overlap = 0.7, eps = 0.7, min_samples = 3
- Graph 4:
  Resolution = 10, Overlap = 0.7, eps = 0.7, min_samples = 3
- Graph 5:
  Resolution = 16, Overlap = 0.7, eps = 0.7, min_samples = 5

The filter function is, as before, projection onto the first principal component. Clustering using DBSCAN with minimum cluster size parameter adjusted per graph.

## 4 Results

Table 2 reports the structural properties of the selected Mapper graphs, while Table 3 summarizes degree distributions.

Graph 1 (Resolution = 12, MinPts = 3) produced 207 nodes and 368 edges across 36 components, with 197 cycles. Graph 3 (Resolution = 16, MinPts = 3) was even larger (232 nodes, 421 edges, 229 cycles), reflecting the finer subdivisions introduced by higher resolution.

Graphs 2 and 5 (MinPts = 5) were smaller, around 100 nodes each, as stricter clustering merged many small clusters. Graph 4 (Resolution = 10, MinPts = 3) fell between these extremes (194 nodes, 337 edges).

Degree distributions confirm these patterns: Graphs 1 and 3 contain many nodes of degree 3–5 with some higher-degree hubs, while Graphs 2 and 5 are simpler and tree-like. Overall, higher resolution and lower MinPts yield more fragmented, cycle-rich graphs, while stricter clustering produces fewer, larger components. These trends highlight the need for statistical testing to separate genuine topological signals from parameter effects.

As for the visual representations of the graphs, see 2a and 2c, as well as the bar plots of their respective node sizes . Note That many of the nodes are relatively small most likely due to the reasons mentioned previously.

| Graph | Nodes | Edges | Conn. comps. | Cycles (len) |
|---|---|---|---|---|
| Graph 1 | 207 | 368 | 36 | 197 |
| Graph 2 | 101 | 187 | 18 | 104 |
| Graph 3 | 232 | 421 | 40 | 229 |
| Graph 4 | 194 | 337 | 36 | 179 |
| Graph 5 | 108 | 200 | 19 | 111 |

**Table 2: Comparison of structural properties across Mapper graphs.**

| Graph | 1–2 | 3–5 | 6–10 | 11–20 | 21+ |
|---|---|---|---|---|---|
| Graph 1 | 9 | 181 | 9 | 2 | 6 |
| Graph 2 | 10 | 82 | 2 | 6 | 1 |
| Graph 3 | 20 | 182 | 20 | 6 | 4 |
| Graph 4 | 10 | 171 | 5 | 3 | 5 |
| Graph 5 | 9 | 84 | 7 | 8 | 0 |

**Table 3: Binned degree distributions across graphs.**

## 5 Figures and Results Visualization

## 6 Discussion

The consistent branching patterns across multiple Mapper graphs suggest genuine topological structure in the repository embedding space rather than parameter artifacts.

The large presence of cycles indicates more complex topological relationships beyond simple clustering, possibly representing repositories that share multiple characteristics or form transition regions between different project types. Although most of these may be attributed to noise. We aim to further explore those that are relevant using the techniques from [7] and [1].

### 6.1 Limitations and Error Analysis

Several limitations must be acknowledged:

*6.1.1 Parameter Sensitivity.* While we observe some consistent patterns across parameter choices, a more systematic exploration of parameter space is needed than a pure grid search.

*6.1.2 Computational Constraints.* Full significance testing of all features is computationally expensive, limiting the scale of analysis possible.

*6.1.3 Interpretation Challenges.* The semantic meaning of topological features requires domain expertise and may not generalize across different types of software projects.

*6.1.4 Embedding Model Dependence.* Results depend on the quality and characteristics of the embedding model used.

## 7 Future Work and Conclusions

### 7.1 Immediate Extensions

- Complete statistical validation of all observed topological features
- Systematic parameter sensitivity analysis
- Comprehensive repository characteristic analysis for interpretation
- Cross-validation with different embedding models and data subsets

### 7.2 Methodological Advances

- Adaptive Mapper guided by significance testing to optimize filter functions
- Validation on simple synthetic datasets to confirm methodology effectiveness
- Development of Mapper quality metrics and automated parameter selection
- Hybrid approaches combining Mapper with other dimensionality reduction techniques

### 7.3 Applications and Validation

- Predictive modeling using topological features for repository characteristics
- Integration with software engineering workflows and tools
- Evaluation by domain experts for practical relevance
- Extension to other software engineering datasets and problems

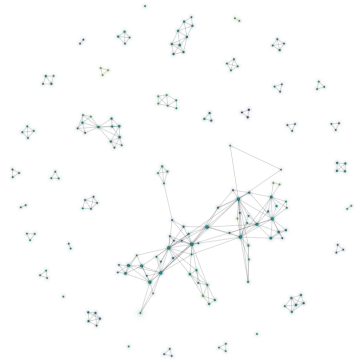
### 7.4 Conclusions

While computational constraints limit the scope of current analysis, the framework establishes a foundation for rigorous topological analysis of software engineering data. The combination of visualization, statistical validation, and manual interpretation provides a comprehensive approach to understanding high-dimensional repository relationships.

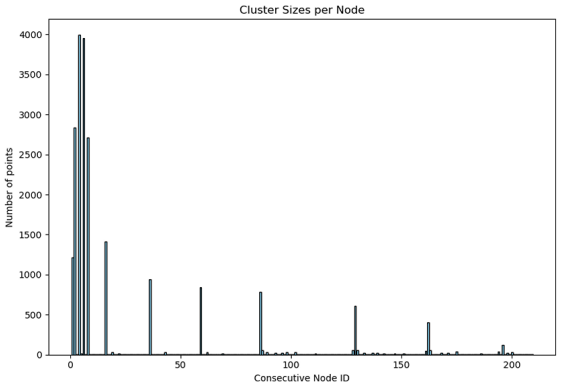
The observed topological structure suggests that repository embeddings capture meaningful relationships beyond simple clustering, opening possibilities for novel applications in software engineering and repository analysis.

## Acknowledgements

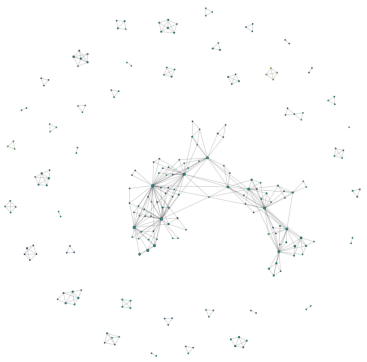
This research was supported by the *EnrichMyData* project, which provided financial support for the work presented in this paper.



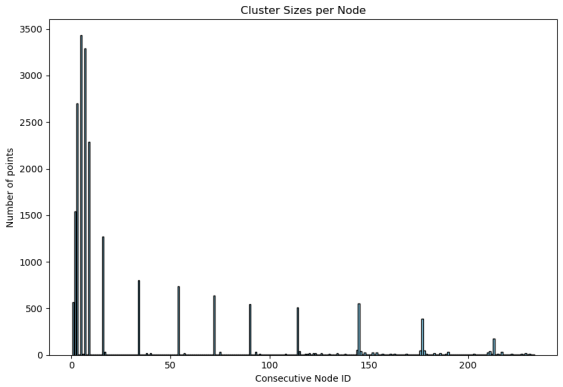
(a) Mapper graph (Graph 1).




(b) Node count distribution (Graph 1).



(c) Mapper graph (Graph 3).




(d) Node count distribution (Graph 3).

**Figure 2: Representative Mapper graphs (Graph 1 and Graph 3) with corresponding node count barplots. Both 2a and 2c show a significant central connected component with some branching, however the boundary of the largest connected component seems to be quite noisy. Further statistical testing will aim to improve upon pruning the noisy artifacts.**

## References


[1] Omer Bobrowski and Primož Skraba. [n. d.] A universal null-distribution for topological data analysis. (). https://www.nature.com/articles/s41598-023-37842-2.

[2] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*. AAAI Press, 226–231.

[3] Ziyad Oulhaj, Mathieu Carrière, and Bertrand Michel. 2024. Differentiable mapper for topological optimization of data representation. In *Proceedings of the 41st International Conference on Machine Learning* (Proceedings of Machine Learning Research). Vol. 235. PMLR, 38919–38936. doi:10.48550/arXiv.2402.12854.

[4] Karl Pearson. 1901. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2, 11, 559–572. doi:10.1080/14786440109462720.

[5] Md Rafsan Jani Rokon, Panagiotis Kallis, Michele Castronovo, Alexander Serebrenik, and Alberto Bacchelli. 2021. Repo2vec: repository embeddings for effective similarity search and recommendation. In *Proceedings of the 18th International Conference on Mining Software Repositories (MSR 2021)*, 384–394.

[6] Gurjeet Singh, Facundo Mémoli, and Gunnar E. Carlsson. 2007. Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *Eurographics Symposium on Point-Based Graphics*. Eurographics Association, 91–100. doi:10.2312/SPBG/SPBG07/091-100.

[7] Patrik Zajec. 2023. Towards testing the significance of branching points and cycles in mapper graphs. (2023).

[8] Yu Zhang, Frank F. Xu, Sha Li, Yu Meng, Xuan Wang, Qi Li, and Jiawei Han. 2019. Higitclass: keyword-driven hierarchical classification of github repositories. In *ICDM '19*, 876–885. doi:10.1109/ICDM.2019.00098.