Topological Data Analysis Project

Gus Brown
gabrown02@wm.edu
College of William and Mary
Williamsburg, Virginia, USA

Abstract

This study examines the structure of the ivy directory of the apache ant-ivy java library. First, all classes, methods, and tokens within the library were vectorized and inspected for metadata such as name, location in the file, and its respective file path. Then, these embeddings were analyzed with standard exploratory data analysis tools to assess location, dimensionality, clustering, density, dynamism, and outliers. The findings from this analysis were then compared to the results of topological feature analysis using the Vietoris-Rips complex and the Distance-to-Measure filtration on the embeddings for each level of code granularity. The results suggest that topological feature analysis on java code structure is significant and could lead to further improvements in code bug detection and assistance.

Keywords: topological data analysis, persistent homology, topological descriptors, persistence diagrams

ACM Reference Format:
Gus Brown. 2024. Topological Data Analysis Project. In . ACM, New York, NY, USA, 4 pages.

1 Introduction

The project objective was to compute and analyze the topological features of a real-world data set. To do so, code fragment embeddings were extracted from the ivy directory of the ant-ivy java library. The Tree-sitter library was used to create syntax trees and parse the ivy library by granularity. For each granularity, these code fragments were then passed into a unixcoder device, a word-to-vector model that transforms each set of code fragments at each granularity into embedding clouds. To obtain any other relevant information, embeddings were fused with metadata such as class, method, token names, the span of bytes where the fragments are located, and the file path which were all stored in a JSON file. To facilitate analytical analysis of these embeddings, these JSON files were stored in three directory structures identical to that of the original codebase: one for classes, one for methods, and one for files.

With the data separated in this way, analytical analysis was performed for each granularity. Before examining topological features, it was important to undergo standard statistical analysis of the high dimensional data. This included assessing location, which was done by examining the L2-norms of the embedding clouds for each granularity. Dimensionality was assessed with Scree-plot analysis, which compares eigenvalues to different features to determine feature importance, and Principal Component Analysis(PCA), which was used to identify how many features account for most of the variance in the data. Clustering and density were assessed with hierarchical clustering and pairwise distance computation. Dynamism was assessed with L2-norm distributions of identical classes, tokens, and methods that were differently located depending on context. Lastly, outlier detection and visualization was done by graphic dimensionality-reduction using T-SNE(t-distributed stochastic neighbor embedding), PCA, and determining outliers with PCA reverse transformation. Topological data analysis reveals high dimensional manifolds that typically remain hidden in standard, dimensionality-reduced analysis. Therefore, it was important to identify features with standard exploratory data analysis first in order to reveal the enhanced insights that topological data analysis has to offer.

To explore the high-dimensional structure and deeper patterns lost in the use of dimension-reduction models for exploratory data analysis, topological data analysis was performed for the embedding clouds of the method and class granularities using the Vietoris-Rips Complex and the Distance-to-Measure filtration, an extension of the Rips complex, implemented with the Gudhi library and Vincent Rouvreau's tutorial on DTM filtration. These complexes, by transforming the point clouds into their topologically realized triangulations, start with simplicial complexes that are homotopically equivalent to their respective point clouds. With no edges or faces in typical point clouds, these simplicial complexes do not have any homological significance initially: no holes or connectedness to be examined. However, the Rips complex and DTM filtration thickens these complexes, meaning they place euclidean, topological balls on these points and increase their radius over time. Using a pairwise distance algorithm, more edges and faces are added as these balls are thickened to become closer to one another in a euclidean

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

sense. DTM is a type of weighted Rips complex, thickening euclidean balls based on parameters p and m that vary the thickening of each ball while a Rips complex thickens balls uniformly.

As these complexes are thickened, the homology, the number of holes and connectedness of the entire structure, changes. The homology of a shape is indicated by the betti numbers, which count the number of holes in each dimension by calculating the dimension of the quotient space between the group of cycles over the group of boundaries for each dimension, effectively counting all cycles that are not boundaries as holes in the skeleton, which is the simplicial complex of faces smaller than a certain dimension. If changes in the homology as these complexes thicken persist, that indicates that they are defining topological features on the manifold underlying the point cloud. If they do not, that indicates that they are noise: insignificant byproducts of the thickening that can be neglected when assessing the point-clouds' underlying shape. Persistence of the complex's homology is visualized with a persistence barcode and diagram, which, in a scatterplot and barcode, graph existence of topological features(holes) with respect to t, showing the life and death of each feature. Long barcodes and points that are lower on the x axis(birth) and higher on the y axis(death) indicate lasting topological features.

2 Results

Each granularity showed notable characteristics in the exploratory data analysis. Examining L2-norms to assess location, the class granularity has an approximately normal distribution, indicating that the class structure of ivy is balanced, having no sharp discontinuities. This is substantiated by the PCA visualization of the class granularity, which has only one, very dense cluster and only detected 20/579 classes as outliers, which were two times the standard deviation away from the mean error found in the PCA reverse transformation. The method granularity had a very uniform L2 norm distribution, making sense to have more variability as methods are more in number and vary more compared to the class level. This is also substantiated by the respective PCA and T-SNE 2D visualizations, which were far more sparse and indicated a higher number of outliers compared to the class level in proportion to its total. The token level showed small variance centered around a mean that was very close to the origin. This makes sense, as the number of tokens compared to the overall structure of the library is very concentrated, explaining the low variability across all features. Feature analysis through Scree plot and PCA indicated that while all levels of granularity had over 700 features, identically, a high proportion of their variance could be explained by far fewer (95% by no more than 200 and all elbow points located at less than 10 features). Clustering as explored through pairwise distances were identical in shape to the L2 norm histograms. The dynamism algorithm indicated that method and class embeddings vary greatly depending on context, which makes sense when considering that their word contents likely vary depending on what these methods and classes are being used for.

Topological descriptors indicated no underlying manifold to the point clouds at the class level, but did identify some topological features on the methods level. Persistence diagrams were successfully produced for the Vietoris-Rips complex and the DTM filtration for the method and class levels, showing the life cycles of all zero and one dimensional features. Rips complexes were calculated with a max-edge length of 50, but these complexes produced extreme Betti numbers ([26,3,0] and [122,360,83] respectively), indicating that the resultant shapes were affected by topological noise. Therefore, only complexes parameterized for a max-edge length of 40 were considered. In the Rips complex produced for the class level, there is a vertical line with zero dimensional features that ends with one feature that never dies and and a multitude of one dimensional features that all die and remain close to the line of identity, indicating their short life. The persistence barcode is, as it should be, in accordance with this result, having only one, long, zero-dimensional barcode and no significantly long one-dimensional barcodes. The DTM filtration for the class level is also in agreement with this, having similar persistence diagrams regardless of its algorithmic difference. Although DTM was too algorithmically expensive for the method level, the last Rip's complex at max-edge length 40 for method embedding, in difference with the others, had one lasting feature at the one-dimensional level. This complex produced the betti numbers [1,1], which is homeomorphic to a circle. While this could be more noise, this does suggest a pattern not found in the exploratory data analysis stage, which visualized the point cloud in the second dimension being one massive cluster, resembling no significant underlying manifold. While it is difficult to interpret the circular nature of the point cloud at the method level, this finding does indicate that topological data analysis uncovered a feature that dimensionality-reduction tools did not.

Performance metrics, scalability, and resource usage played a significant role in the realization of these goals. To counter any scalability and resource usage problems encountered in this project, data engineering and analytical engineering were tested on personal computers, but were ultimately applied using high performance, William and Mary lab machines accessed remotely. Retrieving ivy file names by performing an O(n) search through the code library was manageable even for as large a library as Ivy, and took no more than ten minutes. To parse the code library with tree sitter, generate code embeddings with the Unicoder model, and add parsed metadata along with the embedding to a JSON file took approximately two hours, as it was specifically taxing for the unicoder model to vectorize all granularities within

each file in the ivy library. However, having all granularities separated and equivalent in structure to the ivy library facilitated embedding access on the analytical engineering end, allowing all important data and metadata to be extracted in O(n) time using a python os or glob module. Because there were only 579 classes and 5352 methods, all models and algorithms, albeit expensive, were able to be run with their respective point clouds within 30 minutes, besides DTM filtrations, which were the most computationally expensive. However, it is worth noting that Vietoris Rips complex and persistence calculation were also computationally expensive, having approximately O(n^2) time. The same can be said for the pairwise distance algorithm, hierarchical clustering, and T-SNE, which all were equally expensive as the data extraction itself. However, the scree-plot analysis, PCA feature analysis, correlation matrix calculation, and L2-norm algorithm were all relatively inexpensive and ran within minutes for these granularities. However, there were 378,782 tokens, making expensive algorithms take no less than ten minutes each, rendering extremely expensive analysis tools such as topological analysis, pairwise distances, and hierarchical clustering to be virtually impossible.

3 Discussion

There were numerous inefficiencies in handling the analysis of the ivy library at both the data engineering and analytical engineering side. The use of tree-sitter and unixcoder to extract embeddings and important metadata could have been streamlined, as there was no need to extract urls from the library and store them in a JSON file to then extract embeddings from. Instead, with the github, os, and glob module, embeddings and metadata could have been extracted and placed in a correct directory as the ivy library was traversed. While this is not too computationally impairing, it further obscures the data engineering code from being usable for all libraries. That is, the data engineering code accesses the dowload_urls.json file to produce the embeddings, which is specific to the ivy library, rather than being directly given a github library from which embeddings can be generated.

On the analytical engineering side, model and tool selection could have been perfected. Firstly, a better machine to run code would have made the analytical analysis more holistic, as there were many relevant models and tools used that could not be applied to the token and sometimes even the method level. If used, DTM filtrations on the method level and cluster analysis on the token level could have been utilized. At the same time, both cluster/density analysis methods (pairwise distance algorithm and hierarchical clustering) were too computationally expensive and neither were able to compute at the token level in a reasonable amount of time. Instead, a more efficient algorithm for detecting clusters and density should have been used. Or, as seen in these algorithms/models' agreement with the L2-norm histogram, the L2-norm histogram was likely sufficient to assess both location and density.

4 Future Work

There are many potential applications and future work opportunities implied in the success of this study. On a smaller scale, with better machines and more time, topological data analysis and some of the more computationally expensive tools not used at the token level can be applied, providing opportunity to find more high-dimensional patterns within the code structure. For topological feature analysis, future work may also include the use of confidence bands on persistence diagrams to more definitely define the significance of each topological feature. Also, previously alluded to, by generalizing and refactoring the source code from this study, the data engineering and analytical engineering tools used can be extended to other code libraries. Primarily, code structure analysis can be done on more java libraries in apache or other codebases, with the ultimate goal of recognizing a more global topological pattern in the java language. If global patterns are found, such as the one suggested by the [1,1] betti numbers found in the Rips complex for the method point cloud, these patterns can be used to further enhance java debugging and code-assistance tools that rely on using similar patterns to generate and fix code. As well, unixcoder and tree-sitter can be implemented to parse and vectorize other code languages, such as python and C++, making these implications generalizable to all languages where these modules can be implemented.

5 Links

1. https://github.com/gbrown831/TopologyProject
2. https://github.com/microsoft/CodeBERT/tree/master/UniXcoder
3. https://github.com/GUDHI/TDA-tutorial/blob/master/DTM_filtratio
4. https://github.com/GUDHI
5. https://tree-sitter.github.io/tree-sitter/
6. https://support.cs.wm.edu/index.php/specs

6 Conclusion

To conclude, this study suggests that topological feature detection adds significant insight to high-dimensional data that is obscured by dimension-reduction and standard statistical analysis tools that are industry standard. While there was little found or suggested in the point clouds, specifically for the Rips complex on the method level, the topological feature analysis did offer substantial and contradictory insight to the point cloud that two-dimensional visualizations and statistical tools did not. This indicates that topological data analysis, rooted in proven homology, abstract algebra, and linear algebra concepts, can reveal more patterns than the geometrically derived statistics tools that are more rooted in visual perception. Pointedly, this study has implications for code bug detection and coding assistance but more generally,

analysis of all vectorizable data where patterns and shape are important.

References