Evaluation of different clustering algorithms to cluster test cases, in software testing, as high dimensional data points

Cecilia Battinelli
KTH
cbat@kth.se

Valgerdur Tryggvadottir
KTH
vtry@kth.se

Parastu Rahgozar
KTH
parastu@kth.se

Zhiwu Dong
KTH
zhiwu@kth.se

***Abstract*—This report concerns evaluating different clustering approaches to cluster test cases in software testing. Clustering test cases allows detecting dependencies among them and later on prioritizing them in a such way that faulty software deliveries are avoided. As there is no preferred algorithm, this report aims at providing one, comparing two main algorithms: K-Means, the most famous one and centroid-based; and HDBSCAN, hierarchical and able to identify noise. The evaluation is done using different indexes with respect to a ground truth. Whichever the software is, test cases are represented as high-dimensional data points using Doc2Vec algorithms, thus the underlying structure is the same. This allows to assume the extendibility of results for further software testing. Finally, this report aims at contributing to software testing research area.**

I. Introduction

Software testing is the process of verifying that a software is not faulty. This is done testing it, precisely, given a set of tests to run. These tests, namely test cases (TCs), must be performed in a correct order. Specifically, prioritization is a fundamental step and this can be done detecting dependencies among TCs [1]. An effective way to do that, as suggested by literature [2], [3] is through a clustering approach.

A TC describes what the test itself will check and it is written as a paragraph; on a later step, using Doc2Vec algorithm, given the descriptions of TCs relative to a software, each TC is converted into a n-dimensional vector. Whichever the software is, this is the usual procedure. Successively, the dependencies among those vectors must be detected. This is intrinsically an unsupervised learning problem, which strengthens the use of clustering algorithms. In this report, the validity of two clustering methods, namely K-Means ans HDBSCAN, is investigated. The data which was provided by RISE SICS include the feature vectors for each TC and a ground truth (GT) describing the dependencies among these TCs. The data refers to a train at Bombardier.

The aim of this report is to identify a valuable method able to detect dependencies among TCs. This will be done comparing the performance of each clustering method against the provided GT. The GT will not be used to label the TCs so a supervised learning algorithm can be used. The reason for this is that the dependencies between TCs in the GT are specific for this particular software testing (the train at Bombardier) and can therefore not be used to label data for other software testings. Since the aim of the project is to find a method to detect dependencies and be able to reuse the code unsupervised learning will be used. The data provided will therefore be used to investigate the performance of clustering algorithms. As whichever the software is the relative TCs are obtained using a Doc2Vec algorithm in other words, whichever the software is, the underlying data structure is the same the extendibility of results can be assumed. Specifically, if a given clustering algorithm performs considerably better than the others on the data provided, it would be suggested to use the same approach to cluster further TCs specific for another software.

II. Method

A. Data sets

Two data sets were provided by RISE SICS, `dependencies.csv` (638×2) were each row has a dependency pair of TCs (see Table I), the GT, and `vectors.csv` (1748×64) were each row is a feature vector for a TC.

TC	DependsOn
TC0000	TC0001
TC0000	TC0002
TC0001	TC0002
TC0018	TC0019
TC0018	TC0020
⋮	⋮

TABLE I
Sample of Ground Truth

B. Data preparation

Before working with the data some data preparation was done. The preparation considered was normalization, dimensionality reduction and data manipulation (see II-D). Firstly, a few tests were run on the ”raw” data and it was noted that, applying normalization, resulted in no difference in results. This was expected, as looking at the data the value span mostly the same small interval. Presumably, it can be inferred that the data provided were already normalized by RISE. With respect

to dimensionality reduction, it was chosen to not proceed with that. This is more extensively described in the Appendix A.

C. Dependency Assumptions

The following assumptions regarding the dependencies between TCs had to be done, referring to the GT described in Table I:

- Dependencies are bidirectional:

$$\text{If } A \rightarrow B \text{ then } B \rightarrow A \text{ so } A \leftrightarrow B$$

- Dependencies are transitive:

$$\text{If } A \leftrightarrow B \text{ and } B \leftrightarrow C \text{ then } A \leftrightarrow C$$

D. Comparing with Ground Truth

1) Manipulation of Ground Truth: To be able to compare the clustering results to the GT the `dependencies.cvs` had to be manipulated so that each line has the TCs which are dependent according to our assumptions (see Table II).

TC0000	TC0001	TC0002	
TC0018	TC0019	TC0020	TC0021
⋮	⋮	⋮	⋮

TABLE II
SAMPLE OF MANIPULATED GROUND TRUTH

The data manipulation is done with a recursive function. The output of this function is the manipulated GT, specifically a list of lists where each list corresponded to a cluster.

2) Comparison Algorithm: The comparison between the clustering method output against the GT is done computing "True Positive" (TP), "False Negative" (FN), "False Positive" (FP) and "True Negative" (TN). These labels refer to Table III:

Metric	Ground Truth	Algorithm Output
TP	Yes	Yes
FP	No	Yes
TN	No	No
FN	Yes	No

TABLE III
METRICS' SUMMARY. **YES** MEANS THE EXISTENCE OF A DEPENDENCY. **NO** MEANS THE NON-EXISTENCE OF A DEPENDENCY.

Specifically, two sub-algorithms are needed, one to compute TP and FN and another one to compute FP. TN can easily be computed by difference.

- **TP & FN:** The idea is to go through all the dependencies in the manipulated GT and check for each TC that appears in it if it is clustered together to the TCs it should depend on, hence if it has the same label as the ones that are in the same line. If two TCs that are dependent in the manipulated GT have the same cluster label, then the counter for TP is increased by one; in other words, it increases by one for every correctly detected dependent pair. Otherwise, the counter for FN is increased by one.

- **FP:** When it comes to checking the FP two different cases must be taken into account. The first case concerns the TCs in the manipulated GT. In particular, all the TCs in it should be independent from the TCs in it who are not in the same cluster, i.e in the same line. If this is not the case, thus if they are labelled the same, the counter for FP is increased by one. The second case concerns all the TCs that are not in the manipulated GT, hence that are expected to be independent from every other TC, Each pair of these TCs is checked and if they have the same label, thus if they are clustered together, the counter for FP is increased by one.

E. Evaluation Metrics & Indexes

The evaluation metrics from Table III were used to compute the indexes showed in the Table IV. A more detailed description can be seen in the Appendix B.

Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F-measure	$2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$
Rand index	$\frac{TP+TN}{TP+FP+TN+FN}$
Jaccard index	$\frac{TP}{TP+FN+FP}$
MCC	$\frac{(TP \cdot TN - FP \cdot FN)}{\sqrt{(TP+FN)(TP+FP)(TN+FN)(TN+FP)}}$
FDR	$\frac{FP}{TP+FP}$

TABLE IV
EVALUATIVE INDEXES TO VALUE THE GOODNESS OF A CLUSTERING METHOD

F. Clustering Algorithms

The clustering algorithms considered were K-Means and HDBSCAN. These algorithms are described below:

- **K-Means**: One of the most popular clustering algorithms. It is centroid-based and fast. The drawback is that K, the number of clusters, needs to be defined beforehand. In addition the algorithm clusters everything and is therefore not able to detect outliers. [4]

- **HDBSCAN**: A hierarchical clustering algorithm that works well for high dimensional data and is able to detect independent TCs (outliers) [5].

III. RESULTS

Software testing aims at finding bugs and prevent companies from delivering faulty software. As stated, the prioritization of TCs through dependencies is fundamental. It must be noted that assuming two TCs being dependent when they are not does not affect the outcome, whereas assuming being

independent when it is not the case may destroy the outcome. Given this, major results concern the Rand index and the FDR index. The Rand index is not indicative in software testing; considering K-Means and the extreme case where K = 1747 (all TCs are a singletons with the exception of 2 clustered together) the Rand index is $\approx$ 0.99. However, such a clustering is meaningless as it does not provide any information on dependencies but rather represents the most destructive case. The FDR index describes how many of the detected dependencies are not such in the GT. As the FP dependencies will not negatively affect the outcome, it does not provide any discriminant information. However, aiming at having few FP is still relevant as it avoids time-consuming testing, but, as stated above, the problem with FP does not affect the outcome but only implies a time waste.

The following considerations refer to the algorithms' output, respectively K-Means and HDBSCAN, that provided the best results. The parameters and evaluation metrics used can be seen in Table V. For K-Means the number of clusters is 1300 which means that the amount of test cases in each cluster is not high. This indicates that many clusters will only have one test case and hence the number of FP will be quite low. By comparing the evaluation metrics for K-Means and HDBSCAN it can be seen that some of them are higher for K-Means while others are higher for HDBSCAN. The reason for this and a more detailed comparison between the metrics will be discussed in section IV.

	K-Means	HDBSCAN
Metric		eucledian
Min size		3
Sel method		leaf
Alpha		0.75
K	1300	
Number of components	64	64
Number of clusters	1300	119
TP	391	454
FN	1182	1119
TN	1524953	1524605
FP	352	700
Recall	0.2486	0.2886
Precision	0.5262	0.3934
F-measure	0.3377	0.3330
Rand index	0.9990	0.9988
MCC	0.3612	0.3364
Jaccard index	0.2031	0.1997
FDR	0.4738	0.6066
Elapsed time	28.5158 s	9.9156 s

TABLE V
RESULTS FROM K-MEANS AND HDBSCAN

K-Means is not able to detect outliers, i.e. TCs independent from everything else, and the visualized results prove this. The algorithm tries to put each TC into a cluster and hence, the clusters' density is diverse and more intense if compared to HDBSCAN - this can be seen both in the histogram (see Figure 1) and scatter plot (see Figure 2). I Moreover, the t-SNE representations strengthen this inability (see Figure 7 in Appendix C).

The histogram, Figure 3, for HDBSCAN neglects the outliers (the '-1') cluster - its density is considerably larger than other clusters' density so scaling was not meaningful - and represents a more homogeneous clustering if compared to K-Means. This is empowered by the scatter plot, Figure 4, where the most left-hand points represent all the outliers. It can be seen how dense it is as the points resemble a line, in fact $\approx$ 1200 TCs were regarded outliers.

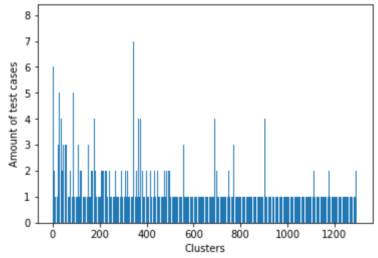

Fig. 1. Shows the amount of test cases within each cluster for K-Means.

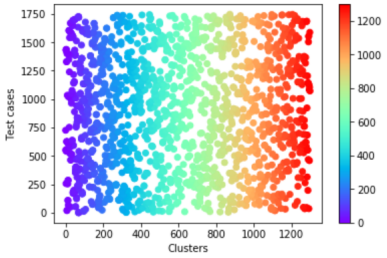

Fig. 2. Shows which test cases are in the same cluster for K-Means.

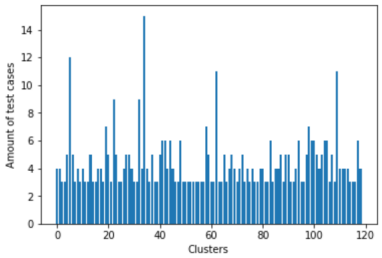

Fig. 3. Shows the amount of test cases within each cluster for HDBSCAN.

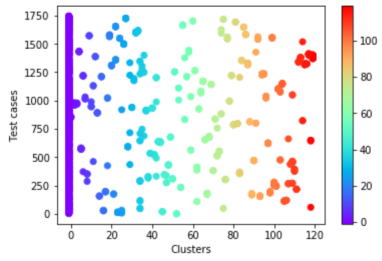

Fig. 4. Shows which test cases are in the same cluster for HDBSCAN.

The more homogeneous clustering is reinforced especially by the t-SNE 2D visualization for HDBSCAN (see Figure

8 in Appendix C), where the prevalent colour (dark blue) corresponds to the outliers. The K-Means t-SNE visualization has many colours and no prevalent one, which corresponds to many small clusters disconnected among each other. Comparing these t-SNE visualization it is clear that HDBSCAN performs better when it comes to detecting outliers.

IV. Discussion

As stated in the Introduction (Section I), in software testing a clustering approach should be preferred to detect dependencies among test cases and finally prioritize them. Indeed, the research topic concerns finding a valuable algorithm able to do that. Results highlighted similar performance for K-Means and HDBSCAN, though intrinsically different algorithms. Firstly, it must be noted that K-Means performed in such a way because a value for K was found in the neighborhood of the optimal one. This was done through a grid search described in the notebook. Whereas, due to the large amount of possible parameters for HDBSCAN, a grid search for HDBSCAN could not have been implemented - it would have resulted in efficiency loss. Instead, a random empirical search for HDBSCAN was done, and, as such, there was no guarantee of optimal neighborhood values. What is noteworthy is that the obtained values of the indexes are close to each other, though different parameters' optimization research was done.

In addition to this, it must be noted that to get useful results for K-Means some a priori knowledge about the structure of TCs should be known. Indeed, if it is known there exist many independent TCs, a large value for K will lead to a decrease in the count of FP, which will increase the value of Precision and eventually provide better F-Measure. In this rare case, K-Means could represent a valid choice as it is one of the fastest algorithm [6] - even though from Table V K-Means elapsed time is larger than HDBSCAN's (this is because K-Means from `scikit-learn` performs a number of iterations as it is sensitive to the start, hence the "absolute" time of one iteration is minor). However, having a priori knowledge is usually not the case. Then, HDBSCAN is able to obtain similar value for F-Measure without using any priori knowledge.

The main difference occurs in the Precision index which is larger in K-Means. However, as it is more important to detect the real underlying dependencies, Recall may play a more important role, and in this case HDBSCAN outperforms. Given this, HDBSCAN performs better than K-Means, as K-Means gets a large value for F-Measure thanks to a large value for Precision. When it comes to MCC index, it can be noted the similarity of the obtained value, which is larger in both cases of F-Measure, underlying the ability of them to detect independencies as well. Jaccard index scores a larger value for for K-Means because the large number of K guarantees a low score of FP. Given all this, the most important thing to notice is that K-Means reached these values through an ad-hoc optimization, whereas HDBSCAN was able to obtain it without optimization and any a priori knowledge, and still they are very close to each other. This is HDBSCAN's major strength.

Overall, HDBSCAN will perform remarkably good using distances valid over Euclidean space, setting the parameter of minimum cluster size to 2-3 - the natural choice as the dependency feature is "pairwise" (i.e. at least 2 test cases) and using `leaf` as selection method, as it allows the definition of more homogeneous clusters [7]. The goodness of this algorithm was expected, as it could be foreseen from literature [5], and accordingly to this discussion, it is expected to outperform other algorithms as it requires no prior knowledge. Finally, in software testing HDBSCAN would be the recommended algorithm to tackle with the unsupervised learning problem. Finally, further research investigation overlapping clustering, such as Fuzzy C-Means, is suggested, so that the transitivity assumption can be avoided.

References

[1] S. Tahvili et al.,"Functional Dependency Detection for Integration Test Cases," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2018, pp. 207214

[2] A. K. Upadhyay and A. K. Misra, "Prioritizing Test Suites Using Clustering Approach in Software Testing," vol. 2, no. 4, p. 5, 2012.

[3] S. Yoo, M. Harman, P. Tonella, and A. Susi, "Clustering Test Cases to Achieve Effective and Scalable Prioritisation Incorporating Expert Knowledge," in *Proceedings of the Eighteenth International Symposium on Software Testing and Analysis*, New York, NY, USA, 2009, pp. 201212.

[4] G. Seif, "The 5 Clustering Algorithms Data Scientists Need to Know," *Towards Data Science*, 05-Feb-2018. [Online]. Available: https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68. [Accessed: 20-Dec-2018].

[5] S. Tahvili et al., "Cluster-Based Test Scheduling Strategies Using Semantic Relationships between Test Specifications," in *2018 ACM/IEEE 5th International Workshop on Requirements Engineering and Testing*, 2018.

[6] "Benchmarking Performance and Scaling of Python Clustering Algorithms hdbscan 0.8.1 documentation," 2016. [Online]. Available: https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html?fbclid=IwAR2W_oAUemqQFBoLuq7RKH1dOvmdjj5F8P1k0_yHZzAVGBGD09szXEtIOoY. [Accessed: 27-Dec-2018].

[7] "Parameter Selection for HDBSCAN* - hdbscan 0.8.1 documentation."[Online]. Available: https://hdbscan.readthedocs.io/en/latest/parameter_selection.html?fbclid=IwAR1QDh6zqViYE4TBsYy9ohspYkboQHJfpa3ER4XckAAbbENK2TIIUuLyCgo. [Accessed: 02-Jan-2019].

[8] G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning*, 8th Edition 2017, pp. 238-239 Springer

[9] K. P. Shung, "Accuracy, Precision, Recall or F1?," Towards Data Science, 15-Mar-2018. [Online]. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. [Accessed: 20-Dec-2018].

[10] "F1-Score," 16-Nov-2016. [Online]. Available: https://clusteval.sdu.dk/1/clustering_quality_measures/18. [Accessed: 20−Dec−2018].

[11] "Rand Index," 16-Nov-2016. [Online]. Available: https://clusteval.sdu.dk/1/clustering_quality_measures/16. [Accessed: 20-Dec-2018].

[12] "Jaccard Index (R)," 16-Nov-2016. [Online]. Available: https://clusteval.sdu.dk/1/clustering_quality_measures/12?fbclid=IwAR0ZAdfWAv1DX4FcsuGA7EPKN2st8XLMRLj4S5U4KBcj9bmchV1BtZH8Tik. [Accessed: 20-Dec-2018].

[13] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Min.*, vol. 10, Dec. 2017.

[14] "False Discovery Rate," 16-Nov-2016. [Online]. Available: https://clusteval.sdu.dk/1/clustering_quality_measures/14?fbclid=IwAR29q05CGIid0OE73Tk3LVCfOMY13ByP0jQ4vrOSCq3qCsduhJ50XhRQXYE. [Accessed: 27-Dec-2018].

[15] L. van de Maaten and G. Hinton, "Visualizing Data using t-SNE," *J. Mach. Learn. Res.*, vol. 1, pp. 1-48.

APPENDIX A
DIMENSIONALITY REDUCTION

With respect to dimensionality reduction, it is known that dimensionality reduction is a useful tool when p - the number of features - is considerably larger than n - the number of observations [8]. Given this, dimensionality reduction was not considered a useful tool for this research topic. Moreover, to double check the validity of this choice, a heatmap and a cluster-heatmap to better understand the correlation between features were plotted. In fact, if any intense correlation among features is found, it can be inferred there is redundancy among them. These maps can be seen in Figure 5 and 6.

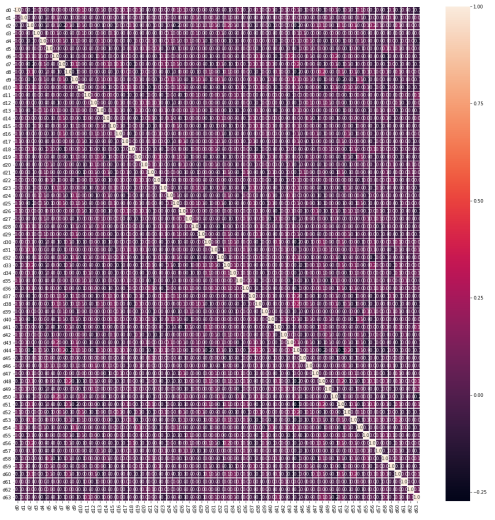

Fig. 5. Heatmap describing the correlation between each pair of features.

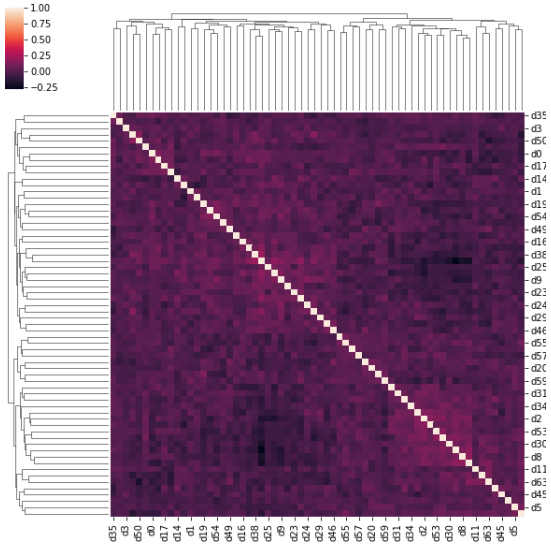

Fig. 6. Cluster-heatmap describing the existing clusters of correlated features.

As can be seen from the figures, the color is always purple (or purplish), with the exception of the diagonal which is white and corresponds to 1.0 correlation. This was expected as each feature is correlated 1.0 to itself. Given all this, it is evident that the features are not correlated among them and no existing cluster of correlated features was detected in the cluster-heatmap. For instance, this proves the non-utility of applying dimensionality reduction in this case.

APPENDIX B
EVALUATION METRICS

The results from the clustering was evaluated by the metrics Precision, Recall, F-measure, Rand index, Jaccard index, Matthew's Correlation Clustering (MCC) and False Discovery Rate (FDR). The formulas for these indexes can be seen in Table IV. The metrics are described below:

- **Precision**: Correctly clustered TCs as a fraction of all the clustered TCs [9].
- **Recall**: Correctly clustered TCs as a fraction of all the TCs that should have been clustered according to the GT [9].
- **F-measure**: Harmonic mean of Precision and Recall [10].
- **Rand index**: Measures the similarity between two clusters. Fraction of TC pairs correctly clustered out of all possible pairs. [11].
- **Jaccard index**: Estimates a likelihood of an element being positive, if it is not correctly classified a negative element [12].
- **Matthew's correlation clustering (MCC)**: Considers equal importance in identifying dependent TCs, as well as identifying independent TCs [13].
- **False discovery rate**: Incorrectly clustered TCs as a fraction of all the clustered TCs [14].

The MCC is particularly valuable as it allows to take into account the TN score without invalidating the index as it happens when using the Rand index. This invalidation of the Rand index is described more extensively in Section III. Moreover, FDR index is taken into account because, even though - as described more extensively in the report - FP dependencies do not represent a major problem as assuming dependent what is independent will not affect the outcome, assuming dependent what is not will result in a more time-consuming testing.

APPENDIX C
VISUALIZATION OF RESULTS

Visualization of high-dimensional data is a challenging task. In this project the t-SNE dimensionality reduction technique jointly to PCA was used to visualize the clusters. This method is popular for visualizing high-dimensional data and was introduced by Van der Maaten and Hinton in 2008. The aim of

this method is to find a representation for the high-dimensional data in a lower-dimensional space. The t-SNE gives each data point a location in a two or three-dimensional map [15]. In Figure 7 and 8 the t-SNE plots for K-Means and HDBSCAN can be seen. By comparing the figures it can be seen that K-Means has many colors and no prevalent one which means that there are many small clusters whereas for HDBSCAN dark blue is the prevalent color and hence corresponds to the outliers. From these figures it can therefore be seen that HDBSCAN is an algorithm which can detect outliers but K-Means is not.

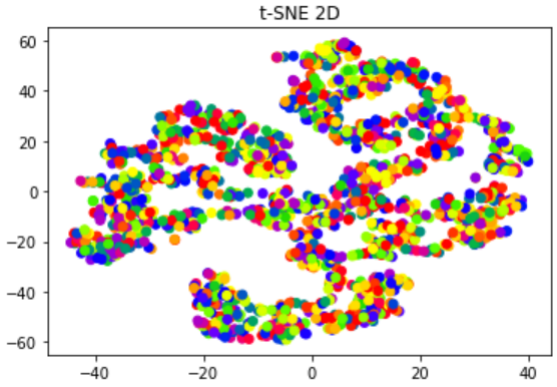

Fig. 7. t-SNE visualization of K-means clustering.

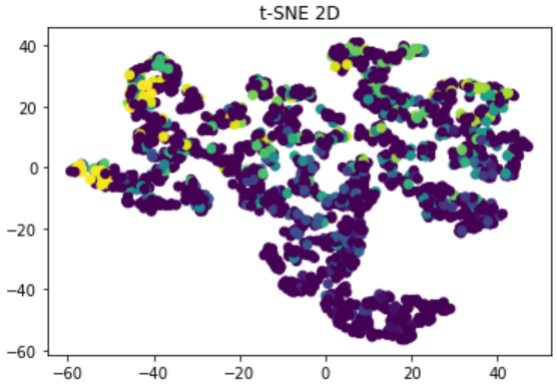

Fig. 8. t-SNE visualization of HDBSCAN clustering.