Project Report: Impact of dimensionality reduction on clustering performance.

Anna Sophia Stein[1]

[1]Department of Linguistics, Düsseldorf

Keywords:
Clustering,
Dimensionality reduction,
Minimum cost flow,
Word embeddings

Abstract Dimensionality reduction techniques are commonly used to reduce the dimensions of data before clustering. High dimensional data usually leads to worse clustering results. However, in the process of reducing the data, some data may get lost which can lead to poor performance of the clustering algorithms. This project aims to investigate the effect of Principal Component Analysis and t-distributed stochastic neighbor embedding on performance of clustering algorithms `k-means` and `DBSCAN`. Results show that `DBSCAN` may be more susceptible to different numbers of dimensions of the input data when its produced by t-SNE. No such effect us found for `k-means`. Overall, the number of `PCA` components (of 95% and 99% variance) do not affect the performance of the clustering algorithms as strongly as the dimension reduction by `t-SNE`does. Further, more detailed research is needed to confirm these findings.

© The Author(s) 2023. Submitted: 03 October 2023 github.com/ansost/NoClu.

1. Introduction

If a project uses high dimensional data for clustering, it is custom to use dimensionality reduction to filter out noise and get more accurate predictions. First of all, it is more difficult to reason about high-dimensional data than two-dimensional data. Technically, most clustering algorithms are based on distance metrics that perform continuously worse with increasing number of dimensions in the data since all points are relatively close together. High-dimensional data can be refactored or projected to a lower-dimensional data set with almost the same information using dimensionality reduction algorithms. However, their usage always leads to some loss in the lower dimensional data compared to the original data.

The goal of this project is to get an initial idea of how the dimensionality reduction techniques affect the performance of clustering algorithms. In particular, I will be using Principal Component Analysis (`PCA`) (1; 2) and t-distributed stochastic neighbor embedding (t-SNE) (3) in combination with the popular clustering algorithms `k-means`(4) and `DBSCAN` (density-based spatial clustering of applications with noise) (5) as implemented by scikit-learn (6). I will be using word embeddings from a pre-trained fastText (7) model as input. Although usually clustering algorithms are unsupervised learning algorithms, using word embeddings enables me to evaluate the clustering performance based on the gold labels derived from the model. The performance is evaluated using a Minimum Cost Maximum Flow (MCMF) algorithm from the Python library `networkx` (8) to compute the overlap between the predicted and the gold clusters.

As the computations behind `t-SNE` are more complex and involve more refactoring of the input data, I expect the `t-SNE`output to have more effects of the performance of the clustering algorithms. Additionally, since `t-SNE` does not preserve the original distances between data points, I expect the cost of the MCMF to vary strongly depending on the dimensions chosen for the input. I furthermore expect DBSCAN, as a density-based clustering algorithm, to perform worse than `k-means`since `t-SNE` is known to produce false, dense patterns where there are none.

My findings suggest that `t-SNE` as a higher impact on the performance of clustering algorithms than `PCA`. Furthermore, my results confirm the hypothesis that density-based clustering algorithms are more susceptible to changes in dimesnionality of the input.

2. Data and Resources

The data used in this project comes from a fastText model which contains word embeddings for Russian noun cases. Additional information comes from metadata files that include a list of all forms, all cases, and a binarily coded matrix to identify syncretic forms. All data is available in the Github repository for this project.

The metadata files were used to filter out syncretic forms from the list of all forms since the filtered data results in clearer clusters. This process resulted in 68,687 individual forms out of an initial 161,210. The resulting list was used to extract the forms' gold labels and vectors. All preprocessing was done using the script `preprocess.py`.

3. Method

The project is split into scripts and source code in the folders `scripts/` and `src` respectively. The folder `src` contains modules the scripts use and only export functions to them. In order to generate output, the scripts are run on the command line. In 1, dotted outlines denote modules and full outlines denote scripts.

The left side of 1 shows the preprocessing of the original input, the first part of which was already explained above in 2. After filtering the data, the first step is to generate different combinations for reducing the dimensions of the input vectors. In order to do that, I used the `PCA` implementation in the `scikit-learn` package to find the optimal number of components for the input data using the cumulative sum of the explained variance in the components. Since there is no clear consensus in the literature on whether to use 99% of variance or 95%, I chose to make two versions of the data, once with each percentage. For the preprocessed vectors, 277 out of 300 initial dimensions explain 99% variance, and 215 components explain 95% of variance. Since 215 components are still too

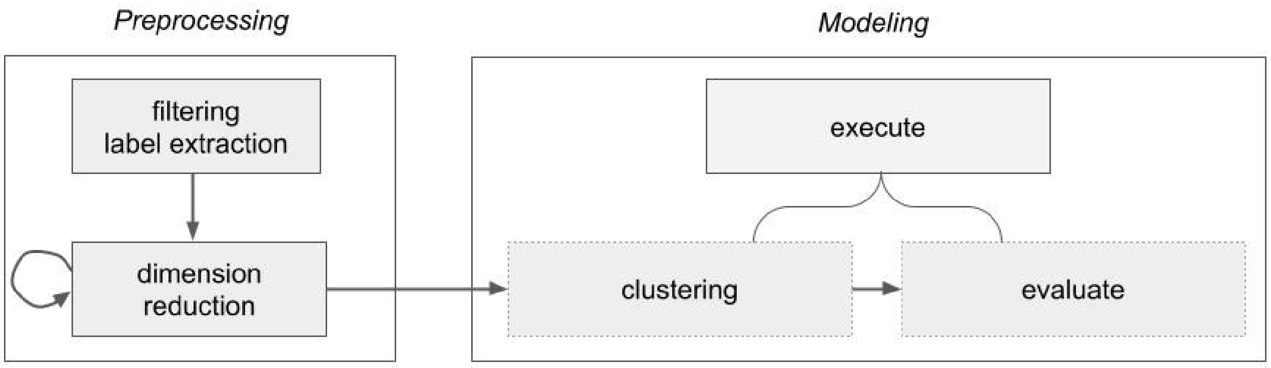

FIGURE 1. Pipeline of the project.

N `PCA` components	**`t-SNE`dimensions**
215	2
215	3
277	2
277	3

TABLE 1. Combinations of dimensionality reduction for the preprocessed input.

many dimensions for most clustering algorithms to perform well on, it makes sense to use `t-SNE` on the output of the `PCA`. This was also done using the `t-SNE` implementation by `scikit-learn`. For t-SNE, there is less room for choosing the number of dimensions. I, therefore, went with the standard parameters of two and three dimensions. This led to four different input versions, as seen in 1. The inputs were produced using the script `reduce.py`, which makes use of the module `pca_tsne.py`. The module only supplies functions for running `PCA` and `t-SNE` as implemented by `scikit-learn`. At the same time, the script produces the variations of input data using combinations supplied in a configuration file.

The lower dimensional input files are then supplied to the script `execute.py`, which makes use of two modules. First, combinations of clustering algorithms and their respective parameters (`epsilon` for DBSCAN, number of clusters to find to k-means) are read from a configuration file, and the functions supplied by the `clustering.py` module are used to compute the labels from the respective clustering algorithm. In order to keep with the scope of this project, I only manipulated the main parameters that control clustering for the clustering algorithms. For `k-means`, this is the number of clusters the algorithm is supposed to find (`n_clusters`). For `DBSCAN`, this is the parameter `epsilon` (ϵ). All other parameters were only modified as necessary or left at their default values (For more information, see the module documentation). Labels that are produced from `clustering.py` are used in the module `evaluate.py` to compute the Minimum Cost Maximum Flow.

The Minimum Cost Maximum Flow algorithm is implemented as a bipartite graph where the predicted clusters (A) and the true labels (B) are two independent sets in which the vertices/nodes represent one cluster/label. The cost of the edges between the nodes in A and B represent the intersection of the shared labels of A and B. For example, if there are 15 labels with *acc.sg.* in the cluster in A, the cost on the edge to the label *acc.sg.* in B would be 15. The cost is represented negatively, so the smaller the number, the better. Accordingly, the cost in the previous example is -15. In addition to the sets A and B, the graph includes a supersink and a supersource that connect to all nodes in A or B, respectively. The MCMF is computed from the super source to the super sink.

For each clustering algorithm, a range of number of clusters/`epsilon` values were tested. They were determined by inspecting the k-nearest neighbors in the original data set for `DBSCAN` and by evaluating cluster inertia for `k-means`. Respectively, the highest rate of change and the point where the inertia starts decreasing linearly is where the optimal number of clusters lies. Since this number is not always easy to find in the plots and there are multiple "elbows" to choose, I chose a range of parameter values to make the development over the different parameter values visible. For `k-means`, a range of 4 to 20 was chosen for the number of clusters, and DBSCAN, an `epsilon` value of 0.5 to 1.8 was chosen.

4. Results and Discussion

4 shows the best MCMF values for each algorithm based on the different input sets. Overall, both algorithms perform reasonably well, given that they are both based on distance measurements within the data points. For both `k-means` and `DBSCAN`, the input with three dimensions leads to worse performance than the two-dimensional input. As initially predicted, `DBSCAN` is more sensitive to changes in `t-SNE` dimensions since it is a density-based algorithm. Between the two- and three-dimensional input to `DBSCAN` lies a difference of 9,318 and 7,136 cost, and even the performance between the input with 255 and 217 components differs significantly as well.

For `k-means`, however, the choice of `PCA` components and `t-SNE` dimensions seems to matter much less compared to `DBSCAN`. Here, the overall difference between the best and worst cost for two- and three-dimensional input is more negligible, although still significant.

This pattern between `k-means` and `DBSCAN` is also visible in the parameters that produced the best clusters. 5 shows the parameters that generate the best output for each clustering algorithm given the input sets. For more straightforward interpretability, the absolute value of the cost is shown instead of the negative value. The number of clusters parameter from `k-means` stays around 20. When comparing the plots for the same number of dimensions but different numbers of components, one can see that the main points where performance drops or increases are very similar for the two-dimensional data. The plots **??** and 3 show the change in performance over the whole range of tested parameters. In both plots, the step from four to five clusters increases performance, and the same

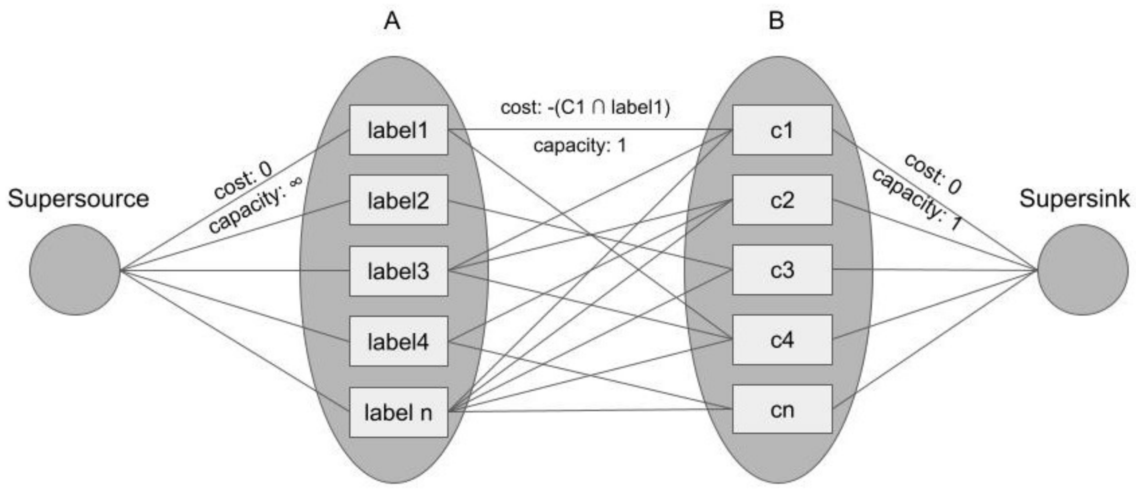

FIGURE 2. Visual representation of the Minimum Cost Maximum Flow algorithm implementation.

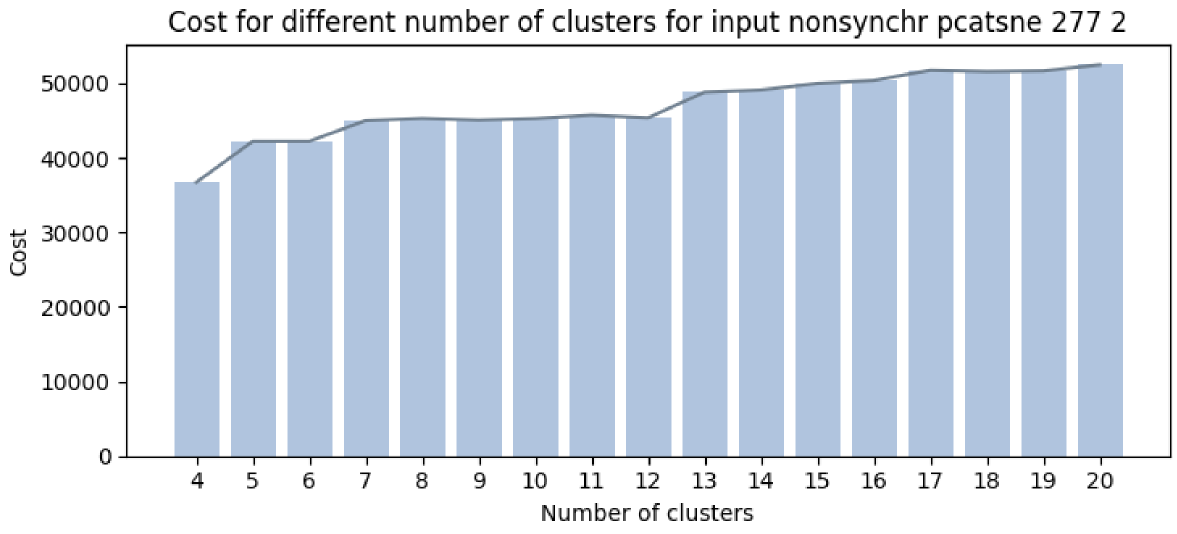

FIGURE 3. Bar plot of costs for labels computed with `k-means` from the dataset with 2 dimensions based on 277 components (95% variance).

happens for the step from 12 to 13 and 16 to 17 clusters. For the three-dimensional data (6 and 4), it seems that the drop in performance for cluster sizes six and nine happens earlier for the data with 277 components, in cluster sizes five and eight.

Epsilon	**Cost**
0.5	-1,444
0.6	-2,429
0.8	-7156
1.0	-15431
1.1	-20368
1.2	-25218
1.5	-37261
1.8	-43681

TABLE 2. Costs for labels computed with `DBSCAN` from the data set with 3 dimensions based on 277 components (99% variance).

The ϵ parameter for `DBSCAN` shows a similar pattern to the algorithm's performance. It is significantly different for the three- and two-dimensional data. This result raises the question of whether the range for ϵ was just not high enough to find its best value. Given that I chose to only test 8 parameters for `DBSCAN` and 16 for `k-means`, it may be the case that one of the parameters in-between or outside of the chosen range would have a better performance for the three-dimensional data. If this is the case, however, it is not apparent from inspecting the k-nearest neighbors for the data set. Tables 2 and 3 show that at least for the input range chosen for this project, $\epsilon = 0.8$ yields the best performance.

Epsilon	**Cost**
0.5	-1,707
0.6	-2,917
0.8	-8,267
1.0	-17,684
1.1	-23,288
1.2	-28,736
1.5	-41,141
1.8	-46,456

TABLE 3. Costs for labels computed with `DBSCAN` from the data set with 3 dimensions based on 215 components (95% variance).

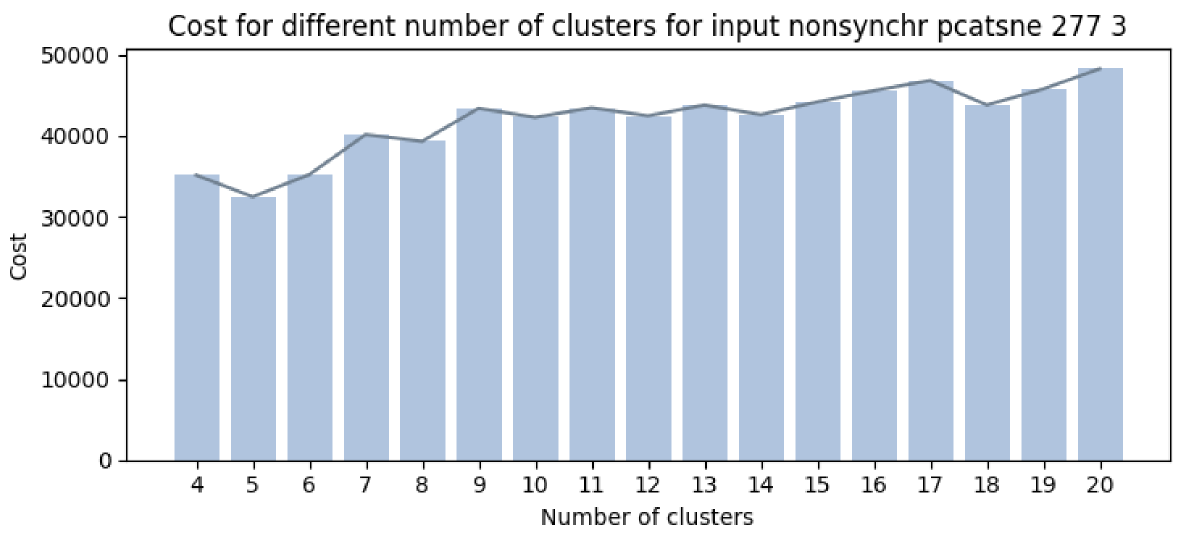

FIGURE 4. Bar plot of costs for labels computed with `k-means` from the data set with 3 dimensions based on 277 components (95% variance).

`PCA` **comps.,** `t-SNE`**dims.**	**k-means**	**DBSCAN**
215, 2	-55,774	-55,326
215, 3	-52,272	-46,456
277, 2	-52,463	-50,844
277, 3	-48,264	-43,681

TABLE 4. The best output per input for each clustering algorithm. The first column lists the number of `PCA` components of the input followed by the number of dimensions used for the `t-SNE`. The second and third column show the best cost flow for the given algorithm.

`PCA` **comps.,** `t-SNE`**dims.**	**N clusters**	ϵ
215, 2	20	0.8
215, 3	19	1.8
277, 2	20	0.8
277, 3	20	1.8

TABLE 5. The best output per input for each clustering algorithm. The first column lists the number of `PCA` components of the input followed by the number of dimensions used for the `t-SNE`. The second and third column show the best cost flow for the given algorithm.

5. Challenges and open issues

Generally, this project would also benefit from a more diverse range of clustering algorithms beyond `k-means` and `DBSCAN`. A hierarchical clustering algorithm (agglomerative clustering with ward) was initially part of the project. However, the algorithm proved to be too memory-intensive (O(nš)) and too time intensive (O(nş) run-time) to run on a data set with 68,687 observations. Incorporating this algorithm in the future requires using the high-performance computing cluster.

Looking back, experimenting with a broader range of values for ϵ would have benefited the analysis. Future analyses, including this, could better investigate whether performance for the three-dimensional data improves and get a more even comparison to the `k-mean` results.

For future research, it would also be interesting to compare this project's results with non-distance-based clustering methods like Finite Mixture Models (9). Non-distance-based clustering algorithms and a broader range of parameter values for `DBSCAN` could both contribute to investigating whether `k-means` is, in fact, more robust against false cluster produced by `t-SNE` or whether this effect results from the nature of `k-means` and the range of ϵ.

Lastly, a significant challenge to this project is the parameter sensitivity of the algorithms versus the parameters modified for this project. Previous research has shown that `t-SNE` is not only very sensitive to its input data but also to parameter tuning for its perplexity parameter (10). Tuning perplexity could significantly affect `t-SNE` output (and accuracy) and, in turn, clustering performance. The same goes for the clustering algorithms, where modifying the default parameters leaves areas to be explored by future research.

6. Summary and conclusion

This project has generated interesting initial results regarding the research question of how dimensionality reduction techniques affect the performance of clustering algorithms. Overall, `k-means` seems more stable against potential `t-SNE` instabilities. However, further research is needed to confirm this finding. `DBSCAN` is much more sensitive to changes in input dimensions, suggesting that density-based clustering algorithms, generally, may be more dependent on the dimensions of the input data. Overall, it seems that `t-SNE` has a more significant impact on the performance of clustering algorithms than PCA. The results of this project could be improved by incorporating a higher parameter variation for the DBSCAN algorithm and exploring the other parameters of t-SNE in particular.

References

[1] Hotelling, Harold: *Analysis of a complex of statistical variables into principal components.* Journal of educational psychology, 24(6):417, 1933.

[2] Pearson, Karl: *Liii. on lines and planes of closest fit to systems of points in space.* The London, Edinburgh, and Dublin philosophical magazine and journal of science, 2(11):559–572, 1901.

[3] Van der Maaten, Laurens and Geoffrey Hinton: *Visualizing data using t-sne.* Journal of machine learning research, 9(11), 2008.

[4] Lloyd, Stuart: *Least squares quantization in pcm.* IEEE transactions on information theory, 28:129–137, 1982.

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

[6] Pedregosa, F., 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: *Scikit-learn: Machine learning in python.* Journal of Machine Learning Research, 12:2825–2830, 2011.

[7] Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov: *Enriching word vectors with subword information.* arXiv preprint arXiv:1607.04606, 2016.

[8] Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart: *Exploring network structure, dynamics, and function using networkx.* In Varoquaux, Gaël, Travis Vaught, and Jarrod Millman (editors): *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[9] McLachlan, Geoffrey J, Sharon X Lee, and Suren I Rathnayake: *Finite mixture models.* Annual review of statistics and its application, 6:355–378, 2019.

[10] Wattenberg, Martin, Fernanda Viégas, and Ian Johnson: *How to use t-sne effectively.* Distill, 1(10):e2, 2016.

Appendix

N clusters	Cost
4.0	-35,312
5.0	-43,524
6.0	-46,971
7.0	-49,767
8.0	-49,453
9.0	-49,838
10.0	-49,198
11.0	-49,886
12.0	-50,274
13.0	-53,092
14.0	-53,558
15.0	-51,961
16.0	-52,831
17.0	-54,989
18.0	-55,097
19.0	-55,150
20.0	-55,774

TABLE 6. Costs for labels computed with `k-means` from the data set with 2 dimensions based on 215 components (95% variance).

N clusters	Cost
4.0	-35,685
5.0	-39,460
6.0	-38,144
7.0	-43,238
8.0	-43,741
9.0	-41,972
10.0	-44,507
11.0	-45,740
12.0	-44,804
13.0	-45,143
14.0	-46,367
15.0	-49,373
16.0	-49,642
17.0	-50,584
18.0	-51,714
19.0	-52,272
20.0	-51,810

TABLE 7. Costs for labels computed with `k-means` from the data set with 3 dimensions based on 215 components (95% variance).

N clusters	Cost
4.0	-35,141
5.0	-32,474
6.0	-35,203
7.0	-40,152
8.0	-39,315
9.0	-43,387
10.0	-42,284
11.0	-43,453
12.0	-42,471
13.0	-43,785
14.0	-42,611
15.0	-44,185
16.0	-45,571
17.0	-46,827
18.0	-43,799
19.0	-45,776
20.0	-48,264

TABLE 8. Costs for labels computed with `k-means` from the dataset with 3 dimensions based on 277 components (99% variance).

N clusters	Cost
4.0	-36,701
5.0	-42,196
6.0	-42,200
7.0	-44,985
8.0	-45,270
9.0	-45,052
10.0	-45,250
11.0	-45,717
12.0	-45,351
13.0	-48,790
14.0	-49,091
15.0	-49,958
16.0	-50,364
17.0	-51,735
18.0	-51,573
19.0	-51,652
20.0	-52,463

TABLE 9. Costs for labels computed with `k-means` from the data set with 2 dimensions based on 277 components (99% variance).

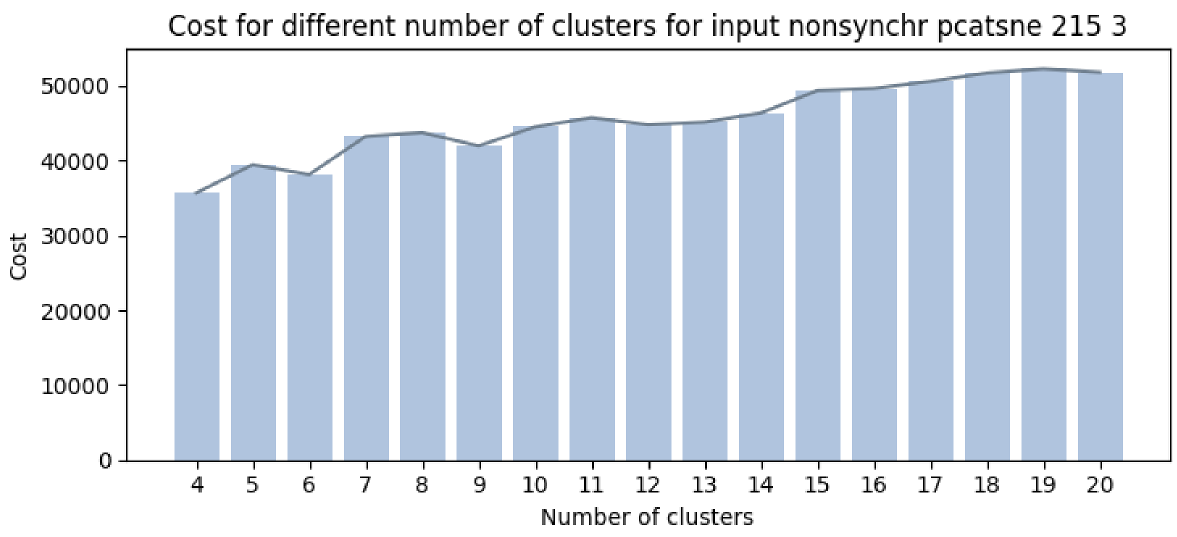

FIGURE 6. Bar plot of costs for labels computed with `k-means`from the data set with 3 dimensions based on 215 components (95% variance).

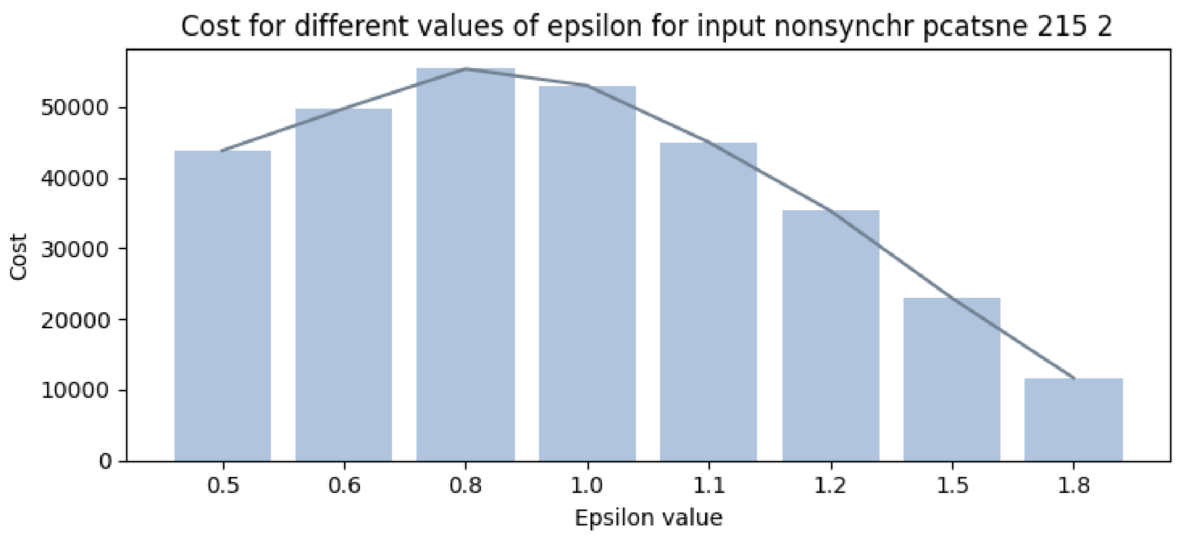

FIGURE 7. Bar plot of costs for labels computed with `DBSCAN` from the data set with 2 dimensions based on 215 components (95% variance).

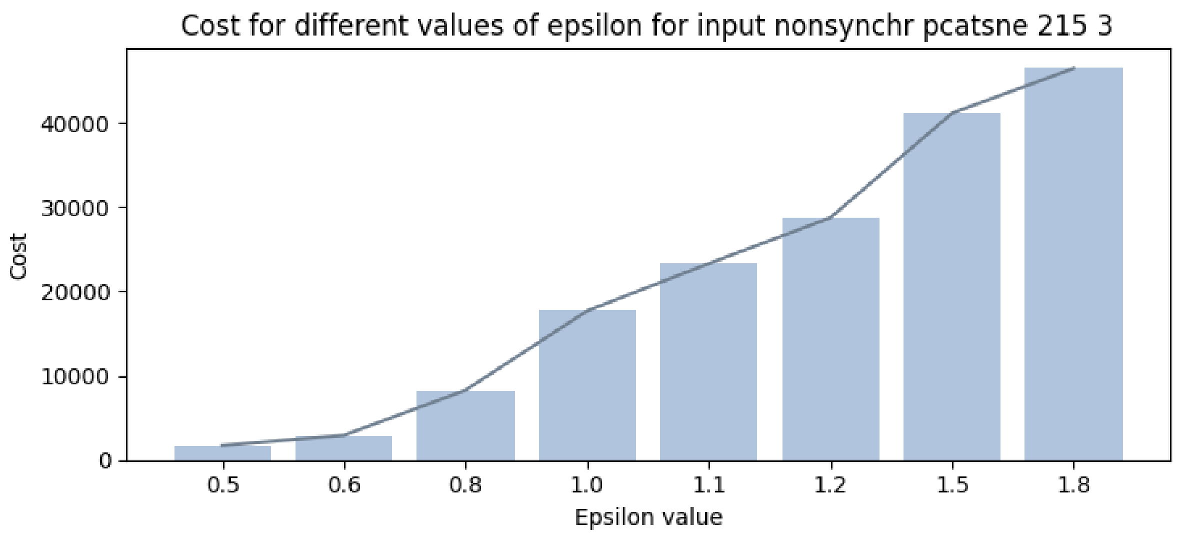

FIGURE 8. Bar plot of costs for labels computed with DBSCAN from the dataset with 3 dimensions based on 215 components (95% variance).

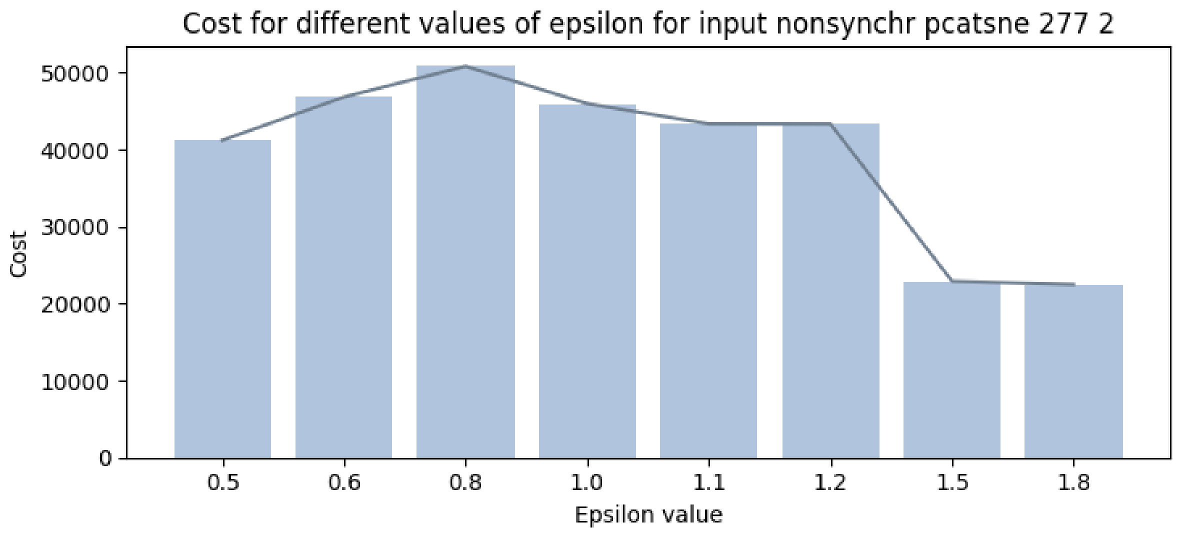

FIGURE 9. Bar plot of costs for labels computed with DBSCAN from the data set with 2 dimensions based on 277 components (95% variance).

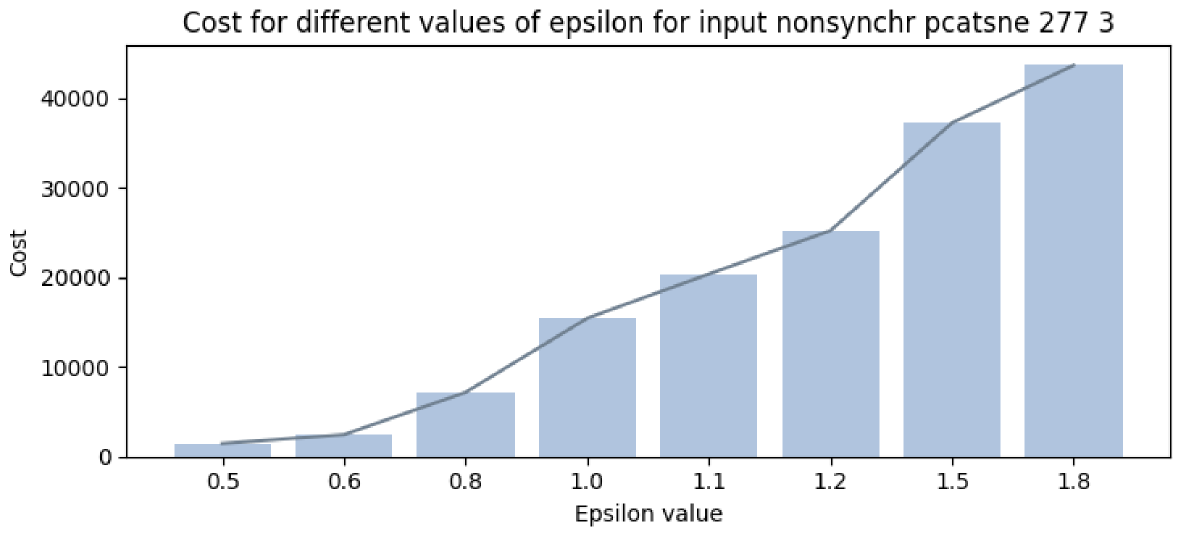

FIGURE 10. Bar plot of costs for labels computed with DBSCAN from the data set with 3 dimensions based on 277 components (95% variance).

Epsilon	Cost
0.5	-43,776
0.6	-49,717
0.8	-55,326
1.0	-52,960
1.1	-44,959
1.2	-35,275
1.5	-22,924
1.8	-11,636

TABLE 10. Costs for labels computed with `DBSCAN` from the data set with 2 dimensions based on 215 components (95% variance).

Epsilon	Cost
0.5	-41,225
0.6	-46,849
0.8	-50,844
1.0	-45,997
1.1	-43,371
1.2	-43,351
1.5	-22,905
1.8	-22,488

TABLE 11. Costs for labels computed with `DBSCAN` from the data set with 2 dimensions based on 277 components (99% variance).

Declaration of Authenticity

I certify that I have written this thesis independently and have not used any sources or aids other than those indicated. All passages of the work, which are taken from other works in terms of wording or meaning, have been marked with the source in each individual case. This also applies to drawings, sketches, sound and video recordings as well as graphical representations.

[illegible]