# Data Mining Assignment - Clustering

Nick Wils

May 18, 2022

**Abstract**

You can find the code of my project with the following link: github_link.
To run the code add the file "articles.tsv" in the same directory as "main.py", or adjust the path manually in the code. You can also look at the github link for a reference. The solution "clusters.tsv", and extra plots will be written in the folder from where you run the code. To test other clustering algorithms, comment the current one out and remove the comment of another one.

## 1 Introduction

In this report I will try to find an accurate way to cluster the given data. The first step in the process is to preprocess the articles. Here the goal of prepossessing is to clean up the data, put it in the same format and remove noise. I took the following steps:

- Remove any non-alphanumeric symbols (except space).
- Put the data in lower case.
- Remove English stop-words.
- Calculate the Term Frequency Inverse Document Frequency (tf-idf) score of the data.

Removing the non-alphanumeric symbols and lower-casing the data is done so that for example the following words count as the same word: Headphone and (headphone). Because, stop-words don't give much context to the data and are just there as noise, it is a good idea to filter them away. With this cleaned up data I calculated the tf-idf score, this is a way to weigh the importance of a word to an article towards the context of the full collection. To calculate this I used a class of Sklearn called "TfidfVectorizer". With this class I also filtered out words that have a document frequency strictly lower than 10 and words that are in at least 90% of the articles. If a word is not used in at least 10 of the 2100 articles, it is a bit too specialized to be used to cluster the articles. On the other hand, if a word is used in most articles it doesn't really distinguish an article.

## 2 Clustering methods seen in class

With the data in a usable state, the next step is to find the right cluster method. I tested all the following methods.

### 2.1 DBSCAN

DBSCAN stands for "Density-based spatial clustering of applications with noise". It uses a distance measurement, Euclidean for example, to group together points that are close together. While doing that it marks outliers in low density regions.

The key parameter that DBSCAN needs is the minimum distance of how close points need to be to be considered a part of the same cluster, epsilon. To find the right epsilon for this dataset I used an existing function that I found here: website. It works by calculating the nearest neighbors of each

point and plotting the result. In the maximum curvature of the plot lies the optimal epsilon. On figure 1 you can see the plot with my data. You can see that the optimal epsilon is 0,01.

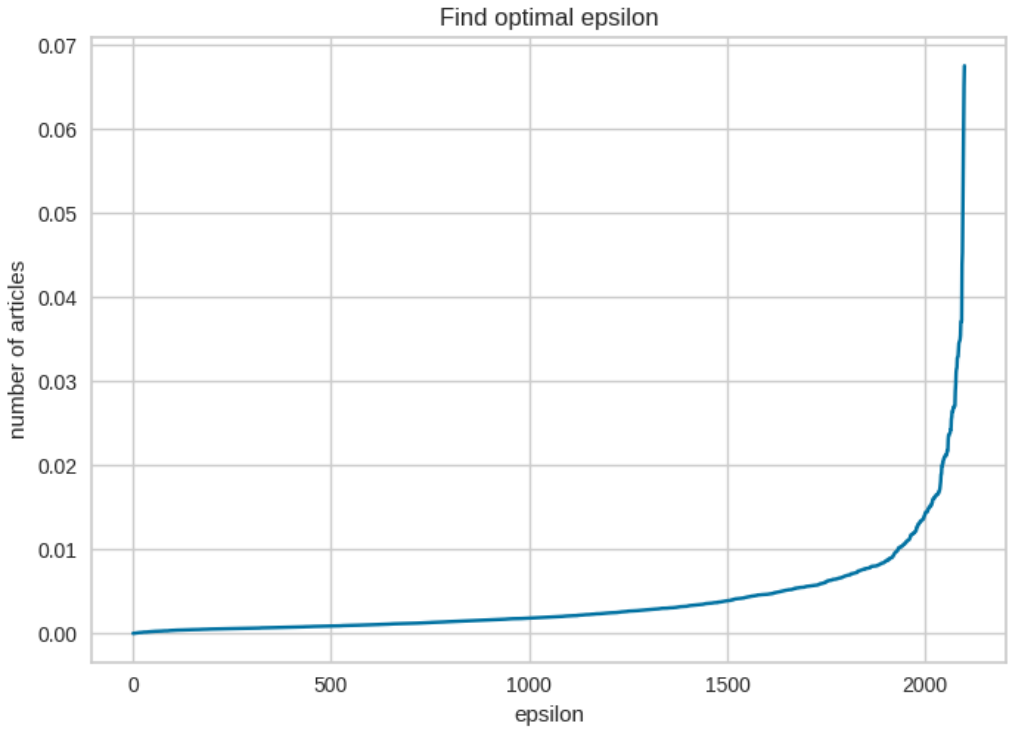


Figure 1: Finding the optimal epsilon for DBSCAN

Theoretically DBSCAN does not sound interesting. The data we have consists of articles so there is a high dimensionality, which is not a place where DBSCAN shines. Running DBSCAN on my data confirmed this. It placed all the articles in the same cluster. I tried this with the following metrics: l2, cosine, euclidean and manhattan.

## 2.2 KMeans

KMeans uses iteration to divide the data into K clusters. It works by first placing K cluster-centers on the data. Each data point belongs to the cluster-center it is closest to. When all the articles are divided, it will re-adjust the cluster-centers so they are in the middle of their cluster. The articles will again be divided to the closest center and so on.

This method is more promising, KMeans is very popular and widely used. The biggest drawback is that it does not work well when the data is shaped in complex shapes. KMeans tries to encircle the data as best as possible, so a spherical shape is ideal. My data is not too complex and I got a good result. Although there were clusters with a low amount of articles in.

## 2.3 KMedoids

KMedoids uses about the same technique as KMeans. The big difference is that KMeans places the center in the middle of the cluster, whereas KMedoids places the center on the closest point to the real center of the cluster. This gives the algorithm more freedom to use different metrics.

This method is not as sensitive to outliers and performed better on the evaluation tests, which I will discuss later. But the differentiating words of the cluster seemed less meaningful than with KMeans.

### 2.4 AgglomerativeClustering

For a hierarchical clustering algorithm I used AgglomerativeClustering [1]. Agglomerative means it has a bottom-up approach. It starts with a lot of very small clusters and one by one it will combine the two closest clusters. The shortest distance between the two clusters can be done in multiple ways. I found that the best result is retrieved by using the "ward" method, which minimizes the variance of the merged clusters.

AgglomerativeClustering has the benefit that it doesn't need to assume how many clusters there are. With my data there was 1 big cluster with 1000+ articles and 12 small clusters, so not an ideal result.

### 2.5 BisectingKMeans

BisectingKMeans is a hybrid between Hierarchical Clustering and KMeans clustering[2]. The algorithm works by starting with one cluster and splitting it up by using KMeans with K=2. It will then calculate the sum of square distance inside each cluster. The cluster with the biggest intra-distance will be the next one that is splitted using KMeans, K=2. It will repeat until K clusters are formed.

The results on the evaluation tests and the meaningfulness of the top differentiating words were similar to KMeans. But BisectingKMeans gave clusters with more similar sizes.

### 2.6 Chosen method

While KMedoids scored by far the best on the evaluation tests, I found that too many clusters differentiated on words like: word, kind, used, usually. Because of this I chose to use the second best method that used more meaningful words to differentiate. That method is BisectingKMeans.

# 3 Finding the number of clusters

## 3.1 KMeans - BisectingKMeans - KMedoids

To find the right amount of cluster for the KMeans, BisectingKMeans and KMedoids methods I used the same two tests.

### 3.1.1 Elbow Test

In this test I calculated the inertia, the sum of squared distances of the articles to the closest cluster center, of multiple K's and plotted the results. Figure 2 shows the plot I got for BisectingKMeans. The goal now is to find the point where the rate, that Inertia decreases, slows down. In my data I did not find an obvious point. I tried looking for even more clusters but the rate does not change drastically. That being said, the rate does slow down. I chose 14 as the point where I found that the rate slowed down significantly enough. This means that 14 would be a good amount of clusters.

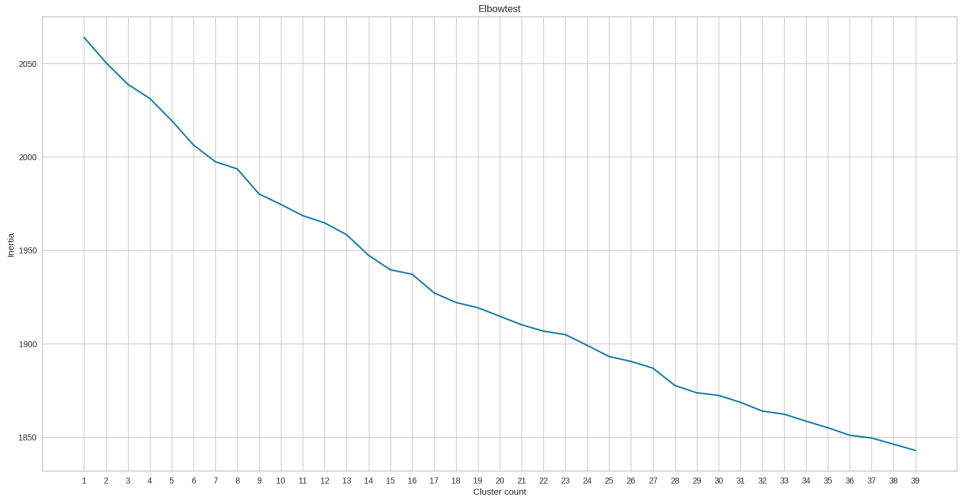


Figure 2: Elbow Test

### 3.1.2 Silhouette analysis

A second analysis used is the silhouette analysis. I plotted the silhouette plot for the method with K around 14 and compared the results.

The silhouette analysis shows the distance between clusters. More specifically, it shows how close points in a cluster are to the closest neighbor cluster. The coefficient it gives back is between -1 and 1. If the coefficient is 1 it means that that point is far away from other clusters. 0 means that that point is on the border of another cluster. -1 means that the point is in the range of a neighbor cluster. Ideally you would want the average coefficient being as high as possible, while avoiding sharp clusters. The silhouette score kept increasing, so it was hard to pick the best one. I chose K=14 again as I found that the shape of the clusters were not too sharp. If a cluster is not sharp, that means that the distance to another cluster is even over the entirety of the cluster. On figure 3 you can see my plot of K=14.

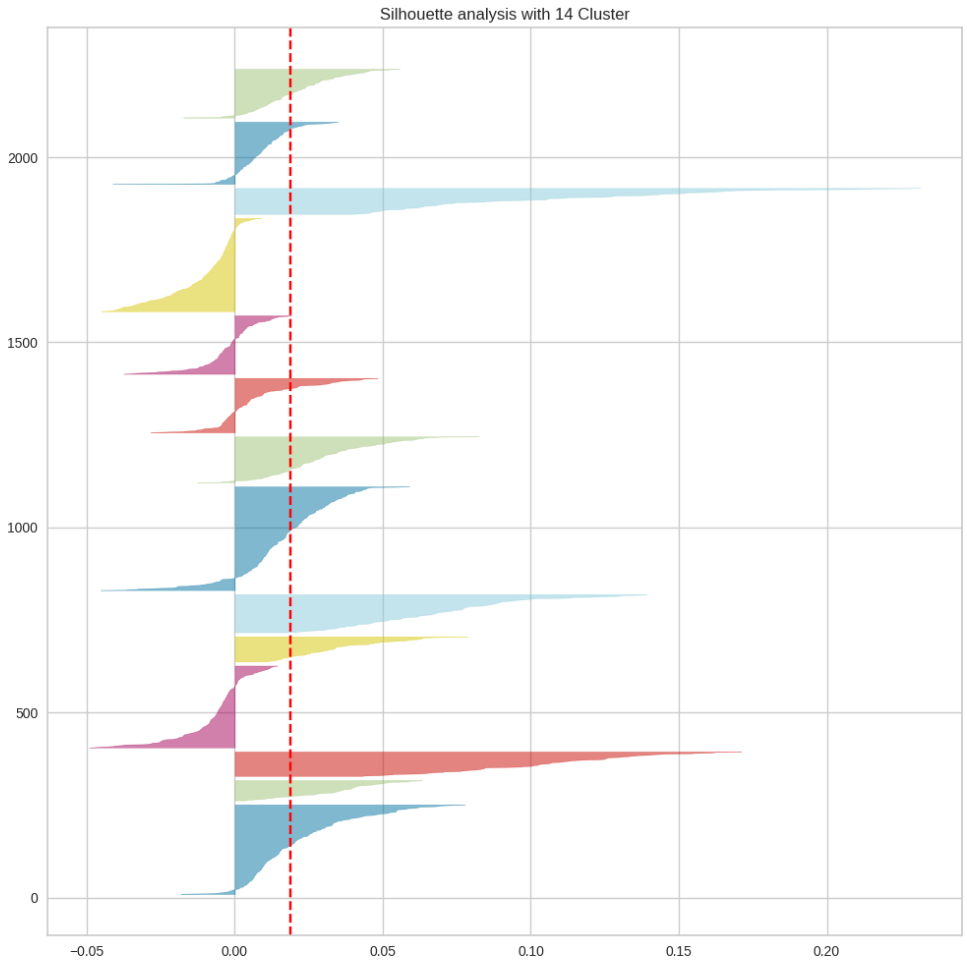


Figure 3: Silhouette analysis

### 3.1.3 Cluster features

The last two things I looked at when evaluating the clusters are: the distribution of the articles in the clusters and the meaningfulness of the top differentiating words. With K=14 I got the following distributions and top 5 differentiating words (words with the highest tf-idf score):

| | # Articles | Top 5 words |
|---|---|---|
| Cluster 0 | 243 | art, architecture, court, legal, law |
| Cluster 1 | 56 | object, geometry, plane, line, point |
| Cluster 2 | 67 | block, encryption, cipher, cryptography, key |
| Cluster 3 | 222 | aviation, used, international, aircraft, television |
| Cluster 4 | 69 | television, money, financial, bank, company |
| Cluster 5 | 104 | current, electricity, electrical, electric, power |
| Cluster 6 | 281 | web, music, internet, software, computer |
| Cluster 7 | 125 | liquid, reaction, atoms, chemistry, chemical |
| Cluster 8 | 148 | person, usually, clothing, worn, people |
| Cluster 9 | 159 | written, ancient, book, art, story |
| Cluster 10 | 253 | person, different, geometry, word, used |
| Cluster 11 | 71 | english, words, word, languages, language |
| Cluster 12 | 169 | behaviour, human, species, psychology, people |
| Cluster 13 | 133 | food, soil, animals, plants, water |

### 3.2 AgglomerativeClustering

For the AgglomerativeClustering I had to use a different method to find an optimal cluster amount. I made a dendrogram, which is a graphical representation of a hierarchical tree. With this dendrogram you can then look up the largest vertical line that does not go through a horizontal line. On this vertical line you then draw a horizontal line and you count how many vertical lines you go through. That's the optimal cluster amount. I did this for my data and got the following figure 4.

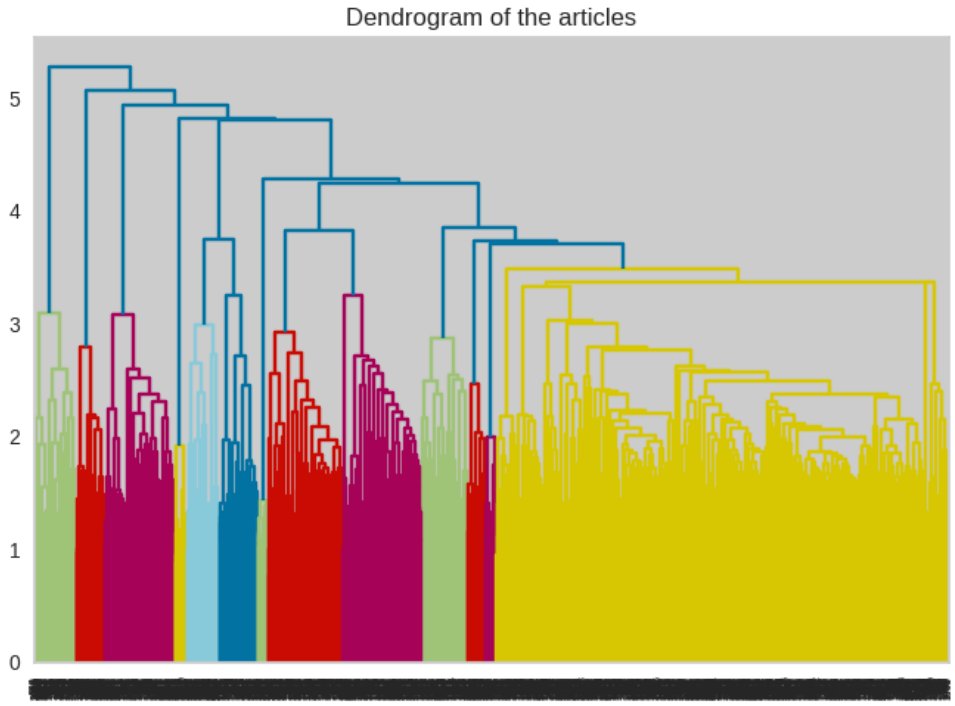


Figure 4: Dendrogram

## 4 Results

With BisectingKMeans as a method and the amount of clusters set to 14, I divided the articles in clusters. There was a lot of subjectivity when making my choices. A simple change such as changing the minimum document frequency from 10 to 25 in the preprocessing, has a large impact on the final result. Furthermore, there is a lot of randomness that comes to play, changing the seed can also have a big impact. I am confident that my clustering is decent. However I do not believe it is optimal. I don't think it is possible to be sure of the optimality of your clustering with a dataset of this size. My results can be found in "clusters.tsv". This is plotted on figure 5.

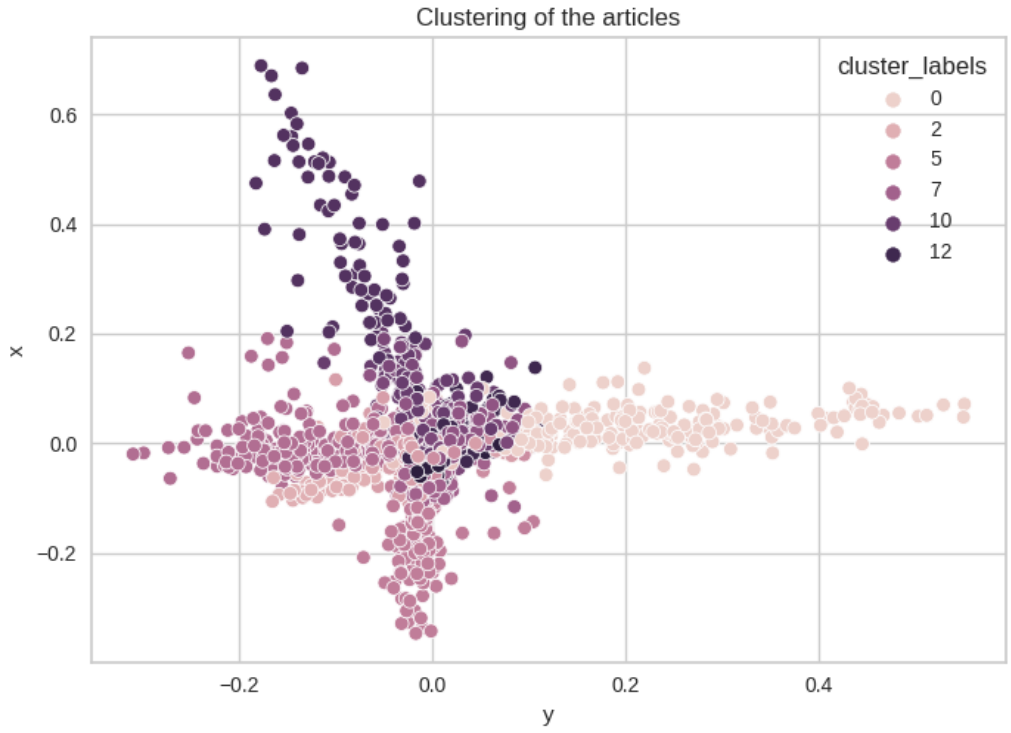


Figure 5: Clustering with BisectingKMeans and 14 clusters

# References

[1] Maklin, C. (2019, July 21). Hierarchical agglomerative clustering algorithm example in Python. Medium. Retrieved May 18, 2022, from https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering- example-in-python-1e18e0075019

[2] Firdaus, A. (2020, May 9). Bisecting Kmeans clustering. Medium. Retrieved May 18, 2022, from https://medium.com/@afrizalfir/bisecting-kmeans-clustering-5bc17603b8a2