Computational Data Science Project

DTU Course 02807

Group 20:
Andreas Kanneworff (s194595)
Farkas Attila Jakab (s230234)
Max Heiberg Bestle (s194574)
Mihai-Valentin Andrei (s222695)
Thor Bjørn Olmedo Gabe(s203861)

28/11/2023

1 Introduction

In our world today, a popular past time for many is to create various Spotify playlists. Each playlist provides an interesting insight into the person making it. With this project, we wanted to examine this playlist data, and see what we could learn about them, using some of the different data science methods we have learned in the course.

For the project, we decided to use a few different methods to analyze the dataset, though we primarily relied on clustering and hashing. We took our dataset, comprised of a million Spotify playlists, and preprocessed it, as the original dataset was very large. Using this preprocessed dataset, we used hashing to see the similarities between the playlists, which we could use for clustering.

2 The Spotify million playlist dataset

2.1 Source of the data

The data we used in this project was obtained from the "Spotify Million Playlist Dataset Challenge".[1]. The dataset consisted of one million public playlists and more than two million unique tracks from 300.000 different artists, and the downloadable zip file was 5.39GB large. The dataset included the names of both the playlists and the tracks within them, the number of albums, the number of followers of the playlist and the duration of all the tracks within it combined. The data was structured in the JSON format and cut into smaller files consisting of 1000 playlists, each sliced file being around 5.5KB large.

In addition to this, in order to find the common traits for playlists we found a dataset describing genre, danceability, duration, energy, instrumentalness, liveliness, loudness, speechiness, tempo, valence, and popularity for each artist, at a total size of 5.22 MB and can be found at Kaggle under "Music Recommendation System using Spotify Dataset". [2]

2.2 Previous analysis

The dataset was used for Spotify's million playlist challenge, which many different people have tried to complete in their own way. Most people try to use deep learning to train a model on the dataset for recommendations. For example, the user victoriab027[3], did some analysis on the dataset, looking into the most popular genres in a playlist. They also looked into the different sounds of the songs, using different elements like acousticness, energy, danceability, etc. They found the averages of each of these elements using all the songs in one playlist.

[1] https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge
[2] https://www.kaggle.com/code/vatsalmavani/music-recommendation-system-using-spotify-dataset/input
[3] https://github.com/victoriab027/MusicDataAnalysis/blob/main/notebooks/1.0-initial-data-exploration.ipynb

2.3 Preprocessing

We quickly realized that a million playlists was too many to work with within our scope of the project. For this reason, we decided to preprocess the dataset, and reduce the size of it. We decided that we would only include playlists, which contained between 50 and 100 tracks, while having at least 5 followers. These conditions gave us the playlists which contained a reasonable number of tracks to apply our chosen methods. Removing playlists with very few followers helped us eliminate the playlist compiled based on extreme interests of just a small number of users. By filtering the million playlists through these criteria, we cut down the number of playlists to 8850, which we decided was a good and manageable number for the data science methods we intended to implement. Thus when filtering the data, we store the playlists that go through the filters in a CSV file, with the following fields: Playlist name, number of tracks in the playlist, number of albums in the playlist, number of followers of the playlist, it's total duration, and the names of all the artists, tracks and albums in it.

This filtered data was then merged with the genre-dataset upon the key 'artists' to create the CSV file, from which to find common traits of the playlists. This was then stored with the following fields: Playlist name, number of followers, number of tracks, artists, genres, acousticness, danceability, duration, energy, instrumentalness, liveness, loudness, speechiness, tempo, valence, popularity, key, mode, and count in a seperate CSV file.

3 Similarity calculation between the playlists

3.1 Description of the method used

The calculation of similarity between the different playlists was done based on the 4th week material of the course. First, we tried to concatenate all the artists, tracks and albums into three longer strings linked to each playlist. The idea was to calculate the similarities between these concatenated strings using different character length shingles, however, we soon realized that this would be an overcomplication of our problem. It is much more convenient to use the unique values of the different attributes as our shingles for the hash calculation. This is possible, because if the same artist, track or album occurs in multiple playlists, the values will exactly match, there are no "similar" attributes referring to the same artist, track of album. Of course, if we merged multiple datasets from different sources, or had typos in the dataset we may have continued with the first approach, but in our case the dataset was generated by Spotify's automated system.

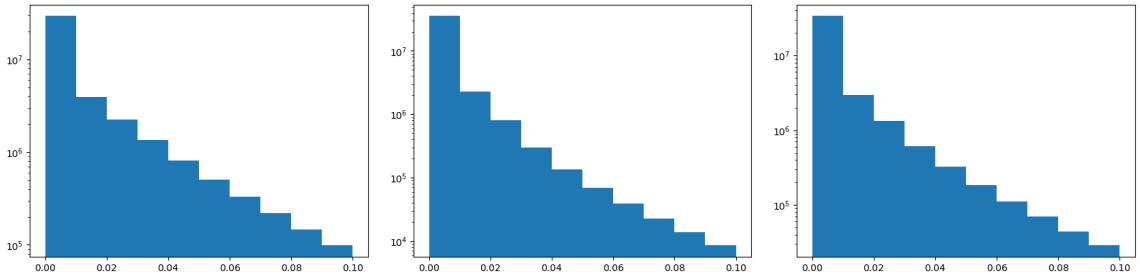
Our algorithm first creates the three shingles (for artists, tracks and albums) for each 8850 playlist as a set of the unique values in a random order. Next, we calculate the belonging minimal hash value based on a number of different seeds to each shingle of the playlists. Lastly, the Jaccard Similarity is calculated between the lists containing the minimal hash values. The calculated values represent a multiple of how many similar items are there in the two compared playlists. The results were stored in in three matrices of 8850 x 8850 similarity values between the playlist. Note, that these matrices are symmetrical and contain only ones in their diagonal. (As all values match if we compare the same playlists.)

We experimented with several ranges of seeds for the minhashing algorithm. We found the optimal value of k=50 to solve our problem. Using a lower number of different seed, the algorithm failed to find all the matching items between the playlists. However, increasing the number of seed above 50, the algorithm have not found more matches (the Jaccard Similarities did not increase anymore), so it did not make sense to use more seeds and more computing capacity to run the program.

3.2 Results

The similarity matrices we created are quite large (there are almost 40 million playlist-pairs that we evaluated) which made it harder to visually represent the output. In order to still present our results, we decided to create histograms which show the distribution of the values in one half triangle of each matrix. (There is no need to include the ones or the same values twice.) Most playlist-pairs had no similar items between them, so an outstandingly large portion of the values are zero. This is followed by the second most common element of 0.01 representing one similar item between the compared two playlists and so on. As there is a magnitude difference between the number of values in the different bins, we display the histograms using a logarithmic scale. Also, we only display the

bins up to 0.1 (10 matching items). The created histograms on Figure 1 indicate a nice geometric distribution of the calculated points.



(a) artists (b) tracks (c) albums

Figure 1: Distribution of the calculated similarity indexes between the playlists

Looking at the histograms we can observe that the artists matrix contains the highest values and the tracks matrix the lowest. This is logical, as the track similarities should describe a subset of the other two similarities. (If a particular track is in two playlists, then the associated artist and album will match as well.)

3.3 Conclusions

Although we realized that artists, track and album titles could have been matched to each other one by one, by using hashing and similarity functions we could greatly reduce the computational power needed for this task. When analyzing the results, we concluded that there are in total less connections between the different playlists than we previously expected. Moving forward with clustering, we decided to only use the similarity matrix created from the artists attributes to have the most amount of edges between the playlists to work with.

4 Clustering

4.1 Results

Next we decided to use the clustering method to analyze the playlist data, especially to better understand the inherent structures or groupings within the 8850 pre-processed filtered playlists.

We first started with a clustering of the playlist by their common songs. For this, we created a dictionary mapping the tracks to playlists and then made a graph in which the nodes are playlists and an edge is added for each 2 playlists that share at least one common track.

To compare the results, we continued by applying the same process to create a clustering of playlists by their common albums and later their common artists. Consequently, we obtained 3 different graph to analyze further.

An interesting discovery of performing this clustering was that regardless of the connection type, there is always one clear dominant clusters that is by far the biggest, followed only by modestly-sized clusters.

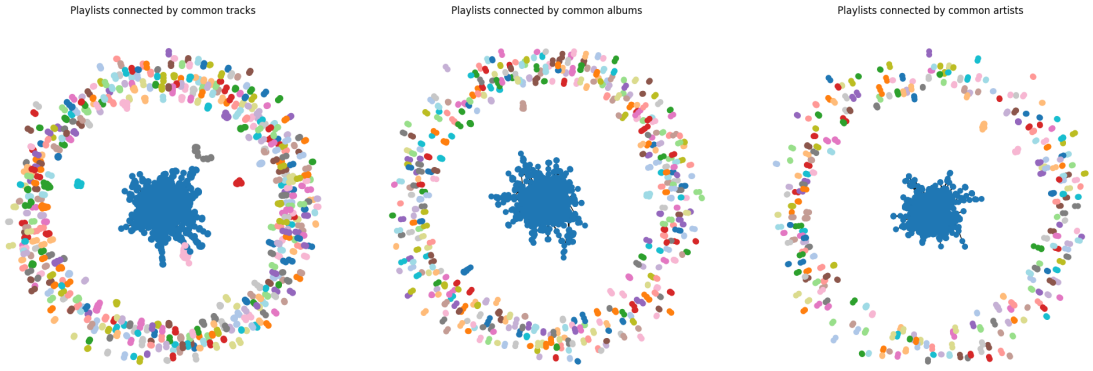


Figure 2: Clustering results of playlists connected by common tracks, albums or artists

Because of the high number of very small clusters, finding an optimal and clear visualization was not straightforward. In order to solve the overlapping of the smaller clusters in the above visualizations we tried adjusting the layout and employing some visualization techniques to help enhance clarity. We used the Kamada-Kawai layout algorithm to achieve an even arrangement of the nodes and a better separation of the clusters and avoid overlapping. The result for this applied to the graph of playlists connected by common tracks can be seen in Figure 3. Because of the very high number of clusters, the layout algorithms does not achieve optimal separation, but it does however clearly show the connections of the largest (blue) cluster.

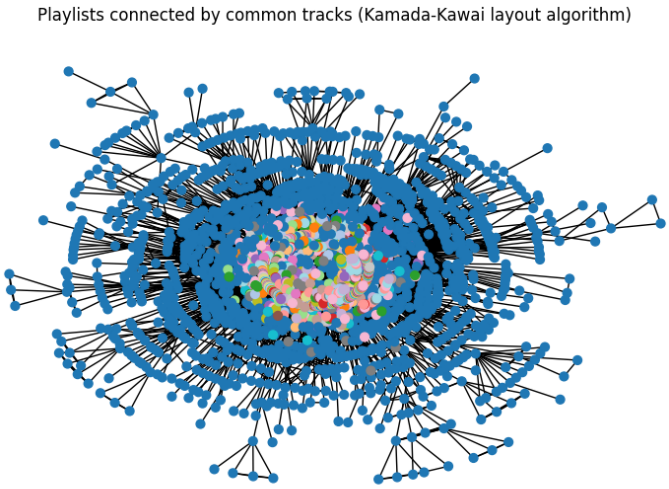


Figure 3: Clustering results of playlists connected by common tracks using the Kamada-Kawai layout algorithm

Next we wanted to better compare the statistics of the 3 different graphs created earlier, by taking a look at the size of the graphs (the number of nodes and number of edges) as well as the size of each of their respective biggest clusters. These results are shown in Figure 4.

As it can be expected, the size of the graph of playlists connected by common artists is considerably larger than the other two, given how more likely it is to find a shared artists between different playlists. In term of the number of nodes, it is increasing from 2823 nodes for shared tracks to 3660 and to 4471. There is however a much higher growth for the number of edges, as there are only 6098 connections by shared tracks, but as much as 40746 connections by shared artists.

As a direct result of this, the first graph has more clusters, they are more separated and the dominant cluster in itself is smaller (454 separated clusters and the biggest consists of 1572 playlists). The albums-graph has 396 connected components with the largest having size 2643, while the last one has only 260 clusters, the biggest of which is 3759 nodes wide. For comparison, the second largest cluster of the first graph has 45 playlists, while the biggest of the third graph has only 11.

Therefore, for the next analysis we will use the graph connected by shared artists, because the larger number of connections will provide more interesting results.

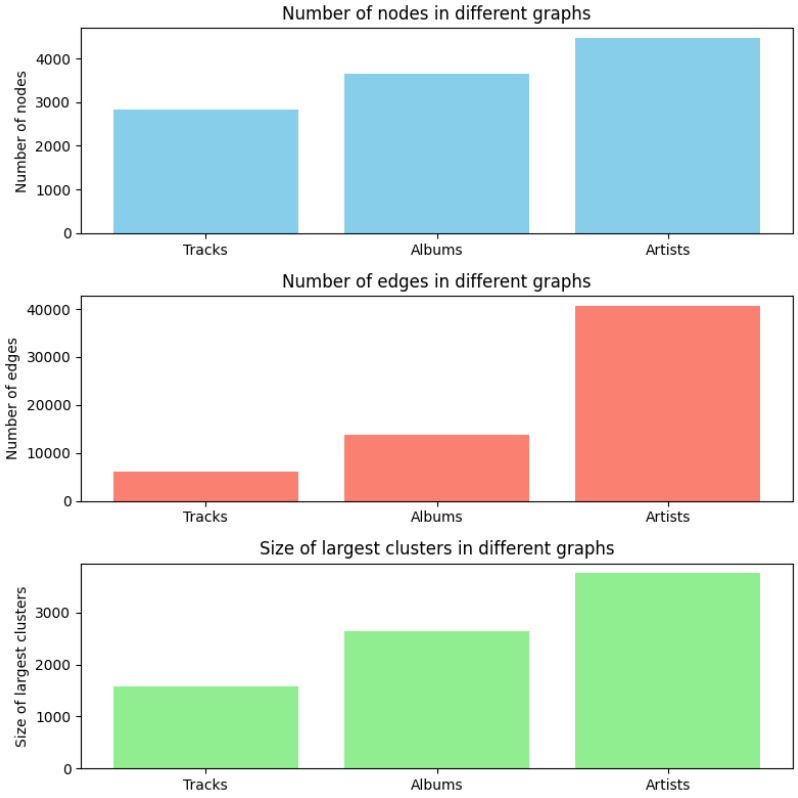


Figure 4: Comparing statistics between the 3 generated graphs

By investigating the largest connected component in this graph, we see that are are a number of 944 different artists that account for the connections. Out of these, by far the most common artist is Drake, whose mentions in the playlist database create 8456 edges in the graph, almost 8 times more than the second most popular singer. The rest of the top 5 artists are: The Weekend (1169 edges), Kanye West, J. Cole and Kendrick Lamar.

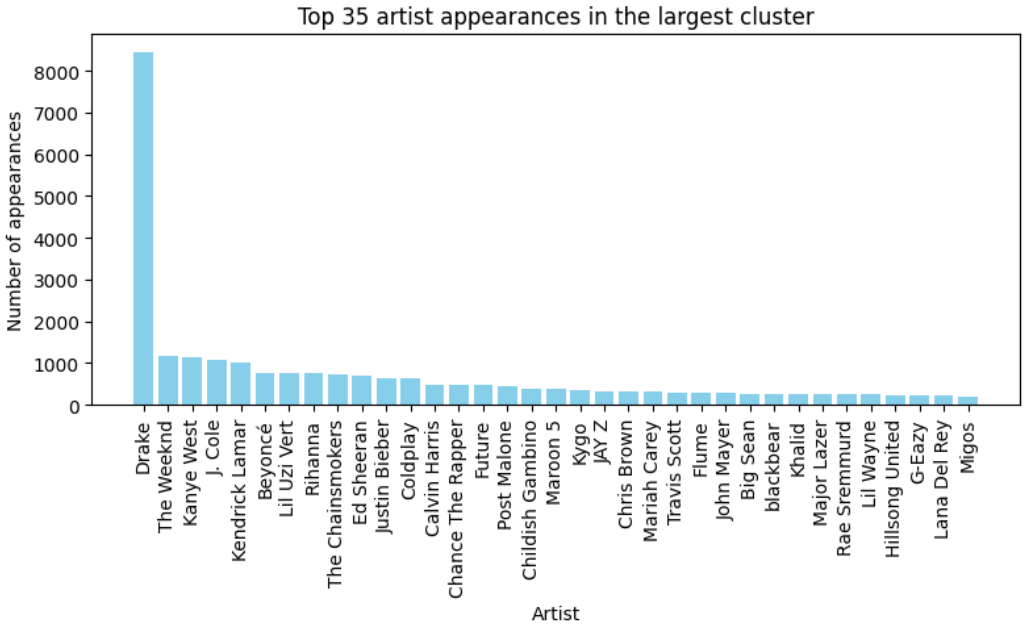


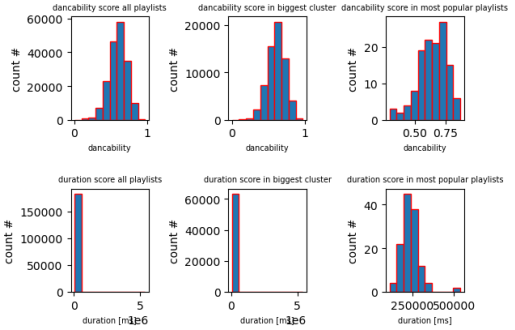
Figure 5: Top 35 artist appearances in the largest cluster

5 Histograms

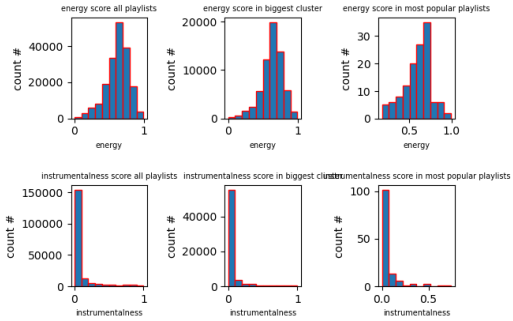
As a topic outside of the course we have chosen data-visualisation through the use of histograms and word-clouds. Information on danceability, duration, energy, instrumentalness, liveliness, loudness, speechiness, tempo, valence, and popularity is available through the dataset, merged upon the artists. It makes it so this analysis effectively is a measure on the artist representation in the playlists and it's effect on the statistics of the playlist. Unless otherwise indicated, these scores are unitless and in the range [0-1] with 1 describing a high presence of the indicated element and 0 describing an absence. Histograms were made using the matplotlib library and is a division of data into discrete bins. We have chosen to bin all data into 10 bins. To capture the most interesting aspects, we set out to test 3 different parts of the dataset; The whole dataset, the largest cluster, and the top 5 most followed playlists.

5.1 Results

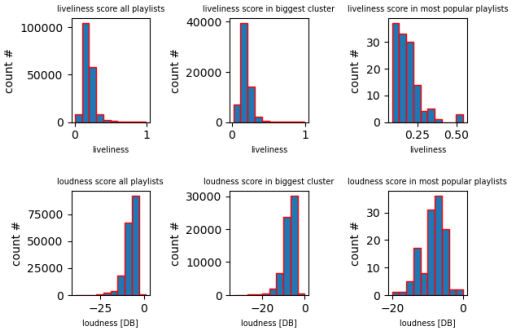
The results for the histograms can be seen in Figure6.



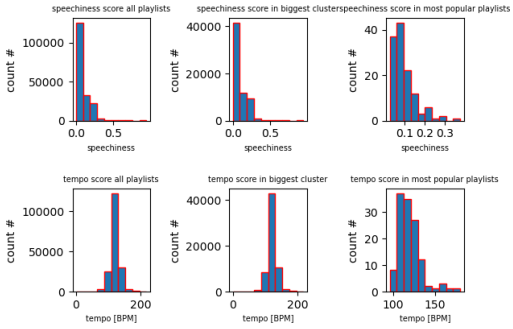
(a) danceability & duration



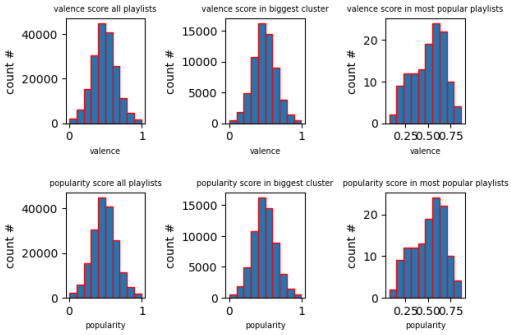
(b) energy & instrumentalness



(c) liveliness & loudness



(d) speechiness & tempo



(e) valence & popularity

Figure 6: Distribution of the calculated similarity indexes between the playlists

5.2 Conclusion

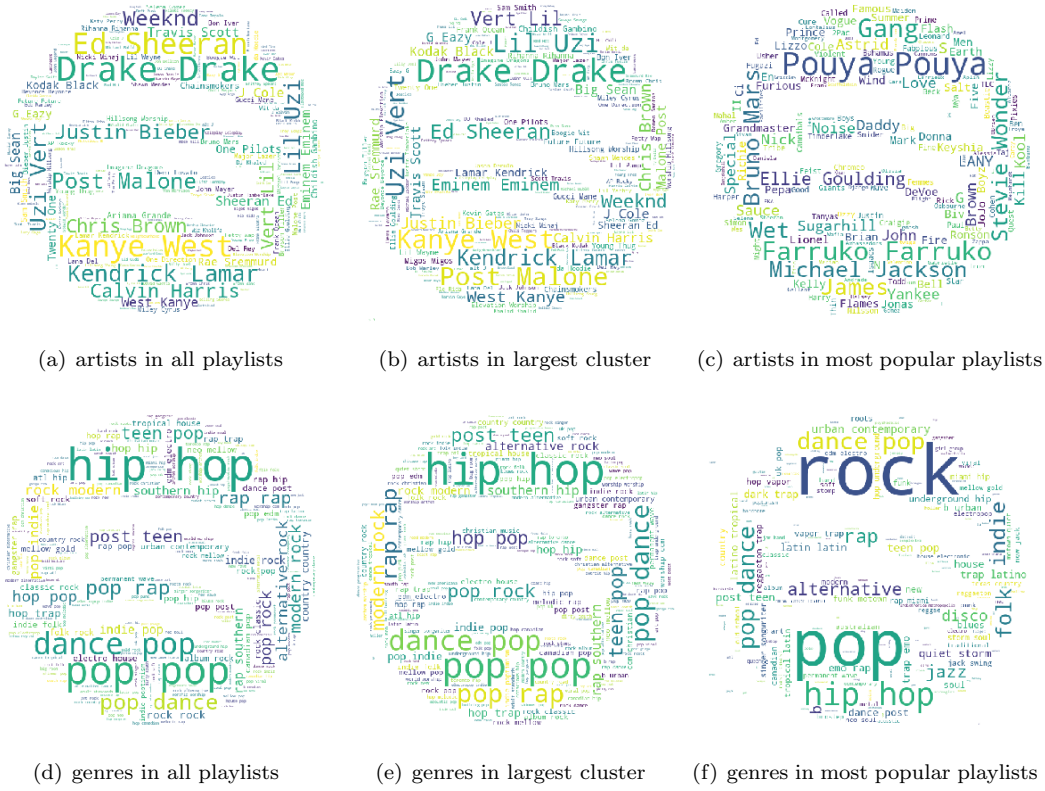
It is clear to see that there is a large overlap between the general data of all playlists and the largest cluster. This is in part because of the sheer size of the cluster, accounting for almost 25% of the total dataset and being clustered based on the most common/popular songs. The most followed playlists differed by tending towards higher danceablity, liveliness, valence, and popularity, and lower loudness compared to the others. Though this could also be due to the much smaller sample size present in the most followed playlists.

6 Word cloud

Word-cloud is a method of visualizing the most common elements in some data through size representation. The more common an element is, the larger it's corresponding text will be. For our implementation we have utilized the wordcloud library. Three different word-clouds were decided upon to find common traits for the data; one for all playlists, one for the largest cluster, and one for the top 5 most followed playlists.

6.1 Results

The results for the word-clouds can be seen in Figure 7.



(a) artists in all playlists (b) artists in largest cluster (c) artists in most popular playlists

(d) genres in all playlists (e) genres in largest cluster (f) genres in most popular playlists

Figure 7: Distribution of the calculated similarity indexes between the playlists

6.2 Conclusion

Looking at the wordclouds for all playlists, largest cluster, and most followed playlists (Figure 7 it is clear to see that there is a large overlap between all playlists and the largest cluster, with the most popular artists (Drake, Kanye West, Kendrick Lamar, etc.) and most popular genres (hip hop, pop, dance pop, etc.) being highly represented in each case. Though, an interesting find was that the most followed playlists deviated from the others by not being overrepresented by the current most popular music and artists. It rather seemed to contain a large amount of 90s and 00s music. When looking at the genre wordcloud for the most popular playlists, a clear affinity for rock seemed to be appear. It could be interesting to test the hypothesis, that playlist follows are not closely

correlated with song popularity in the playlists, and rather tend towards genres such as rock, pop, and nostalgic music from earlier times.

7 Evaluation

Overall, we believe the methods, we chose to apply on our dataset, were good methods for examining the chosen data. Our choice to limit the number of playlists from a million to just less than 10.000, in the preprocessing step, was a good choice, since more data would make some of our figures, like the clustering visualisations, a lot harder to extract knowledge from when viewing them. We believe that our choice of creating word clouds and histograms was a good choice for better visualising the data within the playlists, and fit in with the chosen course methods as well.
The fact that, when clustering the playlists based on songs in common, we get one very large cluster, was at first a surprise. However, when examining it further it began to make sense. This is due to only a few songs, compared to the number of songs created, become very popular, and hence are put in a lot of different playlists. Observations like this were easily made clear using our chosen data analysis methods, which is another reason why we believe we chose correct methods for this specific dataset, and to find the kinds of information we wanted to analyze. Through the analysis using the histograms and wordclouds, we found that the amount of followers a playlist has, might not be closely correlated to the popularity of the songs, but tend more towards certain genres such as rock, pop, or even nostalgic songs from earlier years. This hypothesis could be further tested in future iterations of similar projects.

8 Appendix

8.1 GitHub repository

The notebook containing all of the code as well as the data used can be seen on GitHub. [4]

8.2 Outline of each group member's contribution to the project

- 1 Introduction - **Andreas & Max**
- 2.1 Source of the data - **Andreas**
- 2.2 Previous analysis - **Max**
- 2.3 Preprocessing - **Farkas & Thor**
- 3 Similarity calculation between the playlists - **Farkas**
- 4 Clustering - **Mihai**
- 5 Histograms - **Thor**
- 6 Word cloud - **Thor**
- 7 Evaluation - **Andreas**

[4] https://github.com/jakabfarkas/Computational$_{t}ools_{p}roject$