Visualization of Spotify songs using Schneiderman's mantra

Arvid Magnusson

Abstract— Visualization is a powerful tool in order to understand data. This report describes the implementation of an application used for visualizing Spotify songs using Schneiderman's mantra. It uses three methods to achieve this: a multi-line chart, parallel coordinates and a radar chart. In addition it uses the DBSCAN algorithm to cluster the songs. The final result and evaluation shows that the mantra is a good design philosophy for these types of applications, but that some functionality and interactivity is not necessarily intuitive and easily understandable for users.

1 Introduction

We produce an enormous amount of data nowadays. Visualization is a powerful tool to understand that data. But visualizing it in a good way is not an easy task. The information should be easily interpretable while also giving the correct picture of the data. The application described in this paper aims to use Schneiderman's mantra in order to give a good visualization of the top Spotify songs through the years 2010-2019.

2 Background and Related Work

Schneiderman's mantra, also called the visual information-seeking mantra, summarizes many visual design guidelines and provides a good framework for designing information visualization applications [2]. It states "Overview first, zoom and filter, then details-on-demand". The overview step is meant to give the user a high level view of the data and also act as a starting point for the visualization steps. Zoom and filter is supposed to give the user the ability to explore a certain part of the data by removing unwanted and redundant data and enhancing the interesting part. Details-on-demand allows the user to interact with specific data points in order to get more information about that particular object. In the case of this work, one specific song.

3 Data

The dataset used for the visualization is taken from Kaggle. The dataset consists of 603 songs released during 2010-2019, each containing 14 parameters to work with. The songs are extracted from Spotifys own API. They are the top songs by year according to Spotifys popularity measurement. The parameters consists of both nominal and numerical data. The nominal data types are the song title, artist and the genre. The rest of the parameters are numerical. They describe the characteristics of the song, like tempo, energy, loudness, valence etc.

4 Method

Three different visualization methods were used to cover all steps in the visualization mantra.

4.1 Multi-line graph

The multi-line graph visualizes how the song attributes has changed through the years. It plots the average value for the song attributes each year. It represents the overview part of the visualization mantra. The method was chosen because it is capable of visualizing multiple parameters simultaneously and how they have changed over the years. It also uses a brush over the x-axis which represents the time, in order to filter what data is shown in the parallel coordinates plot. The user can mark which years they are interested in and that data will be sent to the parallel coordinates. The brush enables zooming and filtering of the data, making the other visualization methods easier to work with. The main purpose of the graph is to give the user a starting point in the visualization.

4.2 Parallel coordinates

The parallel coordinates graph visualizes every song from the years selected in the multi-line graph. Each axis represents an attribute value of the song. Parallel coordinates is a good way to visualize high dimensional data in a 2-dimensional way. A drawback of the method is that it can become cluttery really fast if contains a lot of entities from the data. That is why the filtering in the multi-line graph is important. Brushes were also implemented for the parallel coordinates, one brush for each attribute axis. It enables the user to filter specific intervals of the attributes. For example, brushing the tempo axis for values between 150-200 will result in the graph only showing the songs that fulfills that condition.

A drawback of parallel coordinates is that it is only possible to analyse the relationship between adjacent axis. Because of this a modification was done to it, where the user can drag axis and reorder them to their liking. This allows for exploring relationships between all different axis.

Next to the plot a list was implemented that shows all the songs currently visible in the parallel coordinates. The list is continuously updated depending on the filters from the multi-line graph and parallel coordinates.

A hover functionality was also implemented for the plot. Hovering over a line increases the size of it and changes it's color. It also gives a tooltip stating the songs name and artist. In addition, hovering a song in the list will also highlight that corresponding song in the parallel coordinates.

This graph also visualizes the result of the DBSCAN algorithm applied to the data, which is described more below in section 4.4.

4.3 Radar chart

*f*The radar chart represents the final step in Schneiderman's mantra: details-on-demand. It is activated when the user selects a song in the parallel coordinates. It was chosen to visualize the properties of a song. The attributes of the song are shown on the individual axis of the chart. Every time the user selects a new song, the radar chart is reset and re-rendered with the properties of the selected song; it is not rendered together with the previous song. Next to the chart a bit of additional information is displayed: The song name and artist, its duration, the year it was released and its tempo.

4.4 Clustering with DBSCAN

DBSCAN is a density based clustering algorithm that is good at handling noisy data [1]. Given a set of datapoints in some vetorspace, it groups together points that are closely packed together based on density. Datapoints that does not lie in a high density area are marked as noise.

This last property is why this algorithm was chosen for this project over other clustering algorithms like k-means. Its robustness to noise makes it a good method for outlier detection, meaning it is able to find songs that sticks out from the rest. DBSCAN also received the test of time award 2014; an award given to algorithms which have received substantial attention in theory and practice [3].

The algorithm executes when the application is loaded and assigns each point in the data to a cluster/marks it as noise. Each cluster is represented by a color and noise is marked as red. The result of the algorithm is then visualized in the parallel coordinates graph.

5 Implementation

The project was implemented as a web application using HTML/CSS/JavaScript. The main reason for this choice was because of the JavaScript library D3.js. It is an excellent library for creating dynamic and interactive visualizations based on data.

The application was developed in a object-oriented way, with each visualization method being its own class. The classes were then called and created in a main file.

The multi-line graph draws a line for each attribute and plots the average value for that attribute by year. Not every attribute was included in the graph. The nominal data is not included since it cannot be represented this way, and the dB-value was excluded for more practical reasons since it's always a negative value.

The parallel coordinates class was given an update function that is called every time the brush is used in the multi-line graph. The function re-draws the data selected with the brushes.

The same update functionality was given to the radar chart. Every time a song is selected, the update function is called which re-draws the chart with the data from the selected song.

The DBSCAN algorithm was implemented as a function and uses two parameters to determine the clusters: epsilon and minimum number of points. Epsilon specifies the radius of a neighborhood for a point and minimum number of points specifies the required amount of points that should be in the neighborhood. Epsilon was set to 38 and minPoints to 20 since those values managed to capture songs that stood out from the rest.

6 Results

The final application consists of a single page website which mainly uses three methods to visualize the Spotify data: multi-line chart, parallel coordinates and radar chart. The result can be seen in figure 1.

The user starts in the multi-line graph and filters which years to show. The data is then sent to the parallel coordinates. There the user can further filter the data by brushing on the individual axis. A complete list of the currently visualized songs are displayed next to the graph. The user can hover and select individual songs in the graph and list.

The selected song's properties is visualized in the radar chart.

It is not possible to directly compare two or more songs against each other. The data in the radar chart is always reset and re-drawn with the new data.

The multi-line graph does not give very much interesting information. The average value of the attributes changes minimal over the years, making the lines very static. On top of that, the value of the attributes does not say much.

The DBSCAN algorithm was not able to find more than one cluster. Modifying epsilon and minPoints only shifted the size of the noise and the single cluster.

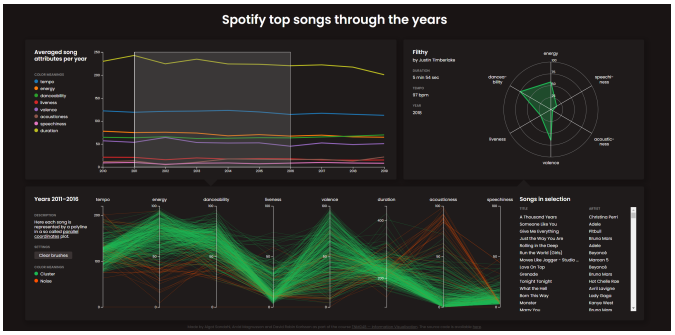

Fig. 1. Snapshot of the final application

7 Evaluation

The evaluation was done with a user-test carried out by three persons. The test-persons first got to solve four tasks while being timed and supervised. These tasks required the person to use all three visualizations and filter methods to solve the task. The information the tester got beforehand was minimal, only some context about the dataset and the attributes. After that, the test-person answered some questions regarding how they experienced the application, where they answered with a number between 1-10, 10 being very easy/good. First there were a few general questions about the application as a whole how they perceived the tasks they got. Then there were two questions per visualization asking how intuitive and usable that particular method was.

Task	User 1	User 2	User 3
Which year did the songs have highest/lowest valence	32 sec	17 sec	20 sec
Find the pattern for most songs attributes	110 sec	110 sec	47 sec
Identify outlying songs and filter out everything except these	160 sec	94 sec	165 sec
What song has highest energy 2013 and how long is it?	45 sec	184 sec	20 sec

Table 1. The four tasks conducted in the user test and the time it took for the users to complete them.

The tasks that involved using the brush function generally was the most challenging for the test-person. As can be seen in table 1, task 1 did not require the use of the brush function and therefore was completed relatively quickly. Task 3 did require the usage of brush and therefore took considerably more time. For task 4, two of the test-persons had gotten a grasp of how the brush worked and could therefore complete it relatively quick.

Regarding the questions part, the test-persons perceived the radar chart to be the easiest one to understand. They thought that parallel coordinates was easy to understand what it represented but the brush functionality was not obvious. The same was said about the multi-line chart. The overall impression of the application was good but the interaction in the visualizations was difficult to grasp in the beginning.

8 Conclusions and Future Work

Schneiderman's mantra is a good design philosophy to follow when visualizing big quantities of data.

Concerning the evaluation part, three test-persons are too few to draw any real conclusions from, by just looking at the data. However, from observing the test-persons trying to solve the tasks, some things became obvious. The brush functionality is not intuitive for a person that has not come across these types of visualizations before. A solution for this could be to implement some kind of indication that the brushes can be interacted with. In addition to that, have a tooltip that says the brushes are interactable.

The result from the DBSCAN algorithm was unexpected. The expectation was that there would be multiple clusters, where each cluster somehow represented a different genre, like pop-song in one cluster and rock songs in another cluster. From the actual result, a conclusion could be that popular song are generally equal to each other. They are characterised by high tempo, energy and danceability. The songs that deviate from this are the occasional acoustic songs, which makes them outliers in the DBSCAN algorithm.

An improvement to the application would be to allow comparison between two songs by drawing them simultaneously in the radar chart. Another functionality would be to have a search bar for songs. Right now it's hard to find a specific song you are looking for. As the application is now, you basically need to know during which year it was released and then scroll through the list to find it.

REFERENCES

[1] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[2] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[3] SIGKDD. 2014 sigkdd test of time award. `https://www.kdd.org/News/view/2014-sigkdd-test-of-time-award`, Aug. 2014.