Towards a Context-Aware Music Recommendation Approach: What is Hidden in the Playlist Name?

Martin Pichl, Eva Zangerle and Günther Specht
Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
firstname.lastname@uibk.ac.at

***Abstract*—New distribution channels like music streaming platforms paved way for making more and more diverse music available to users. Thus, music recommender systems got in the focus of research in academia as well as industry. Collaborative filtering-based recommender systems have been proven useful, but there is space left for improvements by adapting this general approach to better fit to the music recommendations problem. In this work, we incorporate context-based information about the music consumption into the recommendation process. This information is extracted from playlist names, which are analyzed and aggregated into so-called contextual clusters. We find that the listening context plays an important role and thus allows for providing recommendations reaching precision values 33% higher than traditional approaches. Hence, the main contribution of this paper is a new method that extracts and integrates contextual information from playlist names into the recommendation process for improving music recommendations.**

I. Introduction

Recommender systems gained more and more importance in the last decades throughout various fields. Such systems are applied for book, movie, hashtag or even friendship recommendations, just to name a few. In this work, we focus on music recommender systems, as the recommendation of music or rather artists and tracks is nowadays more important than ever: Due to the rise of the web, new distribution channels emerged. Besides online stores like iTunes, streaming platforms like Deezer or Spotify are continually attracting more and more users. Those platforms offer a variety of different music from which the users can choose. In contradiction to traditional channels like the radio, on those platforms users have the free choice, which means they can freely decide when they want to listen to which music. However, this wide variety of different music makes it difficult for the users to find music they want to listen to (at the moment).

As shown in previous research [1], [2], utilizing #nowplaying tweets as a publicly available data source for computing track and artist recommendations is a valuable approach. #nowplaying tweets are tweets, where the users explicitly state to which track they are listening to at the moment on the microblogging platform Twitter. Although additional information about the artist and track is available, e.g., by matching the tweets to a database like MusicBrainz, no or only little information is available about the context of music consumption. However, different contextual information, for instance the geo-location [3], [4], the emotion and mood [5], [6], [7] or a combination of various factors depending on the domain [8] have been proven to be valuable for better performance of music recommendations. Regarding context aware recommender systems, there are two major challenges that will be explained now: First, there is the challenge of gathering contextual information and secondly there is the challenge of integrating this information into the recommendation process. Both challenges are tackled in this work as described in the following.

For achieving the goal of improving collaborative filtering recommender systems as presented in [1], [2], we first took care of challenge 1 by creating an appropriate dataset. We observe that playlist names, amongst others, often contain information about the music consumption context: People create playlists named “party”, “workout”, “my summer playlist”or “christmas”, just to name a few. This finding is congruent with the research by [9] and is the reason why we decided to enrich the #nowplaying dataset by Zangerle et al. [10] with playlist names. In particular, we focus on Spotify users in the dataset, as we can crawl the playlist names and the contained tracks for Spotify users via the Spotify API. Secondly, we focused on challenge 2, how to extract the context. The main idea in this work is to create so-called “contextual clusters”. A contextual cluster is a cluster that aggregates different playlists to a common context. I.e., the “my summer playlist”, “summer 2015 tracks”, “finally summer” and “hot outside” playlists are all clustered in the *summer cluster*. In short, we extract the common listening context, which is hidden in the playlist name. Finally, using this clustering technique, we are able to tag listening events, which consist of a user, a track and an artist with certain contextual clusters. Using this tagged dataset, we show that creating contextual clusters and incorporating them into the recommendation approach is a valuable and promising way for building better music recommender systems.

978-1-4673-8493-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDMW.2015.145

Summing up, the two main contributions of this paper are (i) a method to extract and aggregate contextual information out of playlist names and (ii) an approach to integrate these clusters into the recommender system. We show in the evaluation part of this work that by using the presented methods, we are able to improve the performance of collaborative filtering recommender systems by at least 33%. Additionally, in order to foster research in this field, we publish the playlist dataset on our research website[1].

The remainder of this paper is structured as follows: We briefly present the used dataset and the creation of the dataset in Section II. Afterwards, in Section III, we will focus on the homogenization and aggregation process of the playlist names to context clusters. This is followed by the description of the recommendation approach and the description of the experimental setup in Section IV. The results of the experiments and potential use cases are discussed in Section V, before we finally present our conclusions and an outlook for future research in Section VIII.

II. Dataset

In this section, we introduce the reader to how we create a dataset suitable for context-aware music recommender systems.

For developing a recommender system that utilizes the context of music consumption, we looked for a way to extend the publicly available and up to date #nowplaying dataset by Zangerle et al. [10] [1]. As already mentioned in the introduction, we know that playlist names often contain the desired information. Furthermore, Spotify offers a public API that allows us to request all playlists of a user (in case they are public) along with the contained tracks. As huge parts of the listening events in the #nowplaying dataset are created via Spotify, we decided to combine both sources. The detailed approach is described in the following.

In a first step, we restrict the dataset to users tweeting via Spotify. This can be done, as in the #nowplaying dataset the source is available. The source is the client that was used to publish the tweet. Then we check if a Spotify user exists with the exact same user name. If such a user exists, we retrieve all the playlists of the user including the tracks. In a second step, we validate the user by checking if all the tweeted tracks are contained in the tracks retrieved from Spotify. If this holds, the user is considered as a valid user. We find, that users are tweeting only a small percentage of the tracks which can be found on their playlists. Hence, out dataset is less sparse than the original #nowplaying dataset.

Using this method we generate a "playlist dataset" containing $<user, track, artist, playlist>$-quadruples. In contradiction to the #nowplaying dataset, our dataset is much smaller due to the restrictions, however we find that it is much more dense. It contains 15,345 unique users who listened to 1,878,457 unique tracks by 276,848 unique artists contained in 143,528 unique playlists. On average, each user stores approximately 703 (SD = 2,092) unique tracks in his playlists. From the five point summery of the datasets in Table I we can see that we dramatically increased the number of tracks per user. How valuable this newly created playlist dataset is for research in the field of music recommender systems is presented in the next sections.

[1]available at: http://dbis-nowplaying.uibk.ac.at

Table I
Dataset Comparison: Number of Tracks per User

Dataset	Min.	1^{st} Qu.	Med.	Mean	3^{rd} Qu.	Max.
Playlist	1.0	88.0	334.0	702.8	790.0	186,196.0
#nowplaying	1.0	1.0	1.0	5.94	3.0	84,527.0

III. Cleaning and Contextual Clustering

As already mentioned in the introduction, we want to extract and aggregate contextual information hidden in the playlist names to meaningful information about the user's music consumption behavior. In this section, we describe the whole process in more detail.

As we want to group similar playlists based on their names using clustering, the first logical step is to homogenize playlist names. This is done using lemmatization. Lemmatization is a technique to find the lemma of a given word, which is the base form. We lemmatize our playlist corpus using WordNet[11], a toolkit for natural language processing (NLP). Furthermore, we exclude playlists, which do not contain any additional (context-based) information. These are mainly playlists named after artists, tracks or genres. For this task we rely on AlchemyAPI's entity recognition. AlchemyAPI is a commercial semantic text-analyzing tool. We are using AlchemyAPI, as our experiments showed that its entity recognition works well with respect to tracks, artists and genres. As the (cleaned) playlist names are rather short, creating a meaningful distance matrix suitable for clustering is difficult. We lessen this problem by finding synonyms and hypernyms using WordNet for the lemmas, which have been extracted in the first step. This enables us to create a more expressive term frequency-inverse document frequency (tf-idf) matrix, by using a bag of words describing each playlist based on the derived lemmas, synonyms and hypernyms.

In a next step, we aim to find clusters among the playlists. For finding contextual clusters in the tf-idf matrix, we rely on k-Means clustering. As k-Means requires the parameter k ex-ante as an input variable, the number of playlist groups, or rather clusters k, is determined in the training phase of the recommender system. In the training phase of the recommender system, we compute the within-cluster sum of squares (WCSS) based on the tf-idf matrix as a quality measure for the number of clusters k between $2 \leq k \leq 2 * \sqrt{\frac{n}{2}}$, where n is the number of playlists. This is a valid quality measure, as minimizing the WCSS is the goal of k-Means's objective function [12]. The upper limit

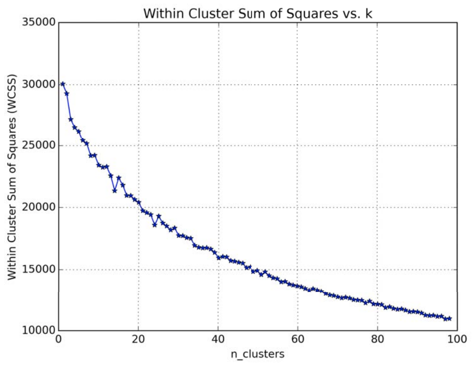

Figure 1. Within Clusters Sum of Squares (WCSS)

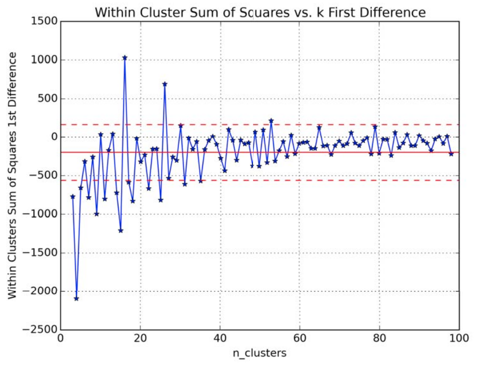

Figure 2. De-Trended Within Clusters Sum of Squares (ΔWCSS)

is based on the approximation rule by [13] for estimating the number clusters k. As we (i) do not recognize the elbow point in the WCSS-curve which can be used as an indicator for the number of cluster k (see Fig. 1) [14] and (ii) aim to determine k numerically, we used the following approach for determining a good k: First, as we know that WCSS declines with the number of clusters, we compute the first order difference (ΔWCSS) to de-trend the WCSS curve [15]. The WCSS-curve is shown in Fig. 1 and the de-trended curve is shown in Fig. 2. The de-trended WCSS curve can be interpreted as the decline/increase of WCSS if k is increased by 1.

In Fig. 2, besides ΔWCSS, also the mean of ΔWCSS, which is subsequently referred to as $\overline{\Delta WCSS}$, is plotted. The dashed lines are the $\overline{\Delta WCSS} + /-$ the standard deviation, which is referred as $\sigma_{\Delta WCSS}$. We determine k by choosing the largest k for which it holds that $\Delta WCSS < \overline{\Delta WCSS} - \sigma_{\Delta WCSS}$. We assume that if the reduction of WCSS is $< \overline{WCSS} - \sigma_{WCSS}$, the clustering performance is not increased significantly by increasing k. This assumption is underpinned by a two-sample Mann-Whitney U test [16] where the *p-value* is smaller than 0.01. Using the dataset described in Section II, the optimal number of clusters was determined as $k = 34$. After presenting our approach of finding and generating meaningful contextual clusters, we show how this approach can be incorporated in a recommender system in the next section.

IV. Recommendation Approach and Experimental Setup

In this section, we introduce the reader to our proposed recommendation approach. We pursue a hybrid approach utilizing collaborative filtering as well as the contextual clusters presented in the preceding section.

As described in Section II, our dataset initially contained $<user, track, artist, playlist>$-quadruples. After the clustering process, we aggregated the playlists to contextual clusters. Thus we have $<user, track, artist, cluster>$-quadruples. As we do not have any content-based information, we rely on collaborative filtering (CF) for computing track recommendations. CF in general recommends items the k-nearest neighbors of a user interacted with, but not the user itself. Allocated to our setting, a CF-based system recommends tracks the nearest neighbors of a user listened to, but not the user itself and thus are new to the user. This concept has already been proven to be useful in such a setting [1], [2]. As the data set does not provide any user ratings on tracks and artists, we use the Jaccard Coefficient [17] in order to determine the user similarity between two users as depicted in Equation 1, where S_i is considered as the set of all tracks the user i listened to. Analogously, we consider S_j as the set of all tracks the user j listened to.

$$Jaccard_{i,j} = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} \qquad (1)$$

If a traditional CF approach would be applied to our dataset, we would compute the similarity of the user we want to retrieve recommendations for to all other users in the dataset. Afterwards, we would recommend items the k-nearest neighbors listened to and are new to the user. However, as we know that various types of context-based information can help to increase the performance of music recommender systems and that the same users are listening to different music in different contexts [5], [6], [7], [8], we apply the presented CF approach separately to each contextual cluster, containing $<user, track, cluster>$-triples. In order to evaluate the performance of the presented recommender system, we conducted an offline evaluation. For this offline evaluation, we split the track listening history of each user into a training- ($S_{training}$) and a testset (S_{test}) by randomly removing $\frac{1}{3}$ of the items. As the track listening histories of the users differ in the size, we decided to make the number of recommendations dependent of the size of the testset. This avoids that users with a larger testset have a higher chance of a hit during the evaluation if the number of recommendations is kept constant. The number of recommendations is computed as depicted in Equation 2.

$$S_{recs} = p * |S_{test}| \qquad (2)$$

During the evaluation the parameter p, which determines the number of recommendations along with the size of the testset, is varied between 0.1 and 1.0. Using this evaluation setup, we are able to compute the *precision* metric as depicted in Equation 3.

$$precision = \frac{|S_{recs} \cap S_{test}|}{p * |S_{test}|} \qquad (3)$$

The numerator of Equation 3 is the number of items recommended which can be also found in the testset and are thus considered as a hit. The denominator is the number of recommended items. Thus, the *precision* is the ratio of hits and recommended items. Besides the *precision*, also the *recall* metric is computed as shown in Equation 4. For computing the *recall*, all items in the testset are considered as relevant. This assumption is made again due to the lack of preference values in the dataset: We have no possibility to distinguish between relevant and irrelevant tracks, i.e., by using a rating or similar.

$$recall = \frac{|S_{recs} \cap S_{test}|}{|S_{test}|} \qquad (4)$$

As for the *precision*, the numerator of Equation 4 is the number of hits. However, in this case the denominator is the number of relevant items – or in other words – the size of the testset. Due to this, if the number of recommendations equals the size of the testset ($p = 1.0$), the *precision* equals the *recall*.

In order to combine both metrics in one number, we combined *precision* and *recall* into the F_β-measure. The exact definition of the F_β-measure is stated in Equation 5. We choose β as low as we put more emphasis towards the precision rather towards the recall. To be precise, as $\beta = 0.1$, we are putting 10 time more emphasis on the *precision* than on the *recall*. This is, as we want to compute a short list of recommendations to the user. According to [18], for this task the *precision* is a suitable metric.

$$F_\beta = (1+\beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall} \qquad (5)$$

The results of the evaluation are presented and discussed in the following two sections.

V. Experiments and Results

In this section, we present the experiments we conduct in this work. We compare three different recommender systems (RS): RS1 (pure CF, no clustering), RS2 (top k-clusters) and RS3 (top 12-clusters, pure CF for the other clusters).

RS1 is a classical CF-based recommender system and thus can be seen as the baseline approach we want to compete. For RS2, we compute and evaluate the average performance

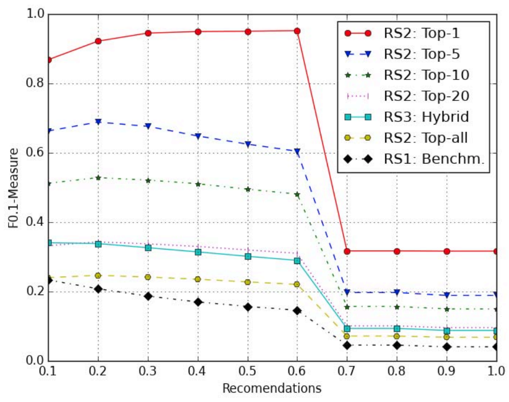

Figure 3. Performance of Different RS

for a different number of top-clusters k. The top-clusters are retrieved by ordering the clusters in descending order by their precomputed performance. As already mentioned in Section IV, this performance is measured using the $F_{0.1}$-measure. The results can be seen in Fig. 3. For RS3, we compute the average performance for all clusters, for which the precomputed $F_{0.1}$-measure is higher than the $F_{0.1}$-measure of the baseline approach. This is combined with the average performance of the minor clusters. As minor clusters, we consider clusters, for which the $F_{0.1}$-measure is lower than the $F_{0.1}$-measure of the baseline approach. For the minor clusters, CF is not applied separately to each cluster. Instead, we merge the minor clusters to one big cluster. This is why RS3 is called a switching recommender [19]. More details why we choose to evaluate this approach are stated in Section VI.

In Fig. 3 we see the $F_{0.1}$-measure computed as stated in Equation 5 for a different number of recommendations. As already explained in Section IV, the length of the user profile and the percentage value p determines the number of recommendations: This implies that the longer the user profile the more recommendations are computed in order to avoid biasing the performance positive with users that have a long user profile.

Independent of the number of recommendations, reaching from 10% ($p = 0.1$) to 100% ($p = 1.0$) of the testset size (see Equation 2), we see that the presented clustering approaches (RS2, RS3) clearly outperform the baseline approach. We can also see that the different clusters have a high variance among their performance: There are contextual clusters that work extraordinarily good, i.e., the clusters among the top-5 and top-10 are performing 3.7 and respectively 2.9 times as good as the baseline approach. However, there are also clusters that perform worse than the baseline approach. This is the reason why the average over all contextual clusters (RS2: top-all) performs 1.3 times better. The average performance over all p values ($\bar{F}_{0.1}$-

Table II
AVERAGE $\bar{F}_{0.1}$-MEASURE OVER ALL p

top-k	$\bar{F}_{0.1}$-measure	$\frac{\bar{F}_{0.1,competitive}}{\bar{F}_{0.1,baseline}}$
top-1	0.6856	5.38
top-5	0.4679	3.67
top-10	0.3664	2.87
top-20	0.2367	1.86
hybrid	0.2275	1.78
all	0.1693	1.33
baseline	0.1275	1.00

measure) is stated in Table II. Our presented approach delivers an at least 33% better performance than the baseline and thus can be considered as a valuable approach. However, the experiments have also shown that not all contextual clusters deliver satisfying performance results. We elaborate on reasons for this in the next section.

VI. Discussion of the Results

In this section, we discuss the evaluated recommender systems of the preceding section in more detail, elaborate on reasons for the different performance of different contextual clusters and finally state potential use cases for a recommender system as the presented one.

The recommender systems RS2 and RS3 have been evaluated after we observed that not all contextual clusters are valuable for improving track recommendations. With RS2, we show how good the different contextual clusters work. The intention behind RS3 is to check the feasibility of what is called a "switching recommender system" in literature [19]: In our setting, such a recommender system utilizes the contextual clusters in case they work good and standard CF as a fallback for contextual clusters not valuable for improving recommendations. For implementing such a switching recommender system, we need to know ex-ante for which contexts to switch. This is an open issue discussed in the as part of future work in Section VIII. In this work, we limit ourselves to investigate what is the reason behind contextual clusters with a unsatisfying performance. We observe two main reasons: First of all, there are clusters clustering playlists names like "…favorites …", "…random …" or "…playlist …" and similar. Those clusters are naturally not valuable for our recommender system, as those clusters do not contain any additional (context-based) information. Secondly, besides those clusters, there are also contextual clusters without a common sense. It seems that there are numerous contexts where people have a common taste, i.e. christmas or summer and there are contexts where the music consumption differs widely. Examples for the latter would be music listened during a workout or during work. An indication for this hypothesis can be seen in Fig. 3: Different clusters have a rather different performance. Furthermore, we computed the Pearson correlation coefficient between the WCSS and the performance measures using the $F_{0.1}$-measure. A low WCSS value means that the clustering for this cluster works good. One could expect that if the clustering works good, the performance of the cluster is good. In this case, the correlation coefficient should be negative (low values for WCSS means high values for the performance). However, this is not the case. The coefficient is 0.15. This can be seen as another evidence that the music homogeneity differs widely among different music listening contexts. Additionally we checked if there is a correlation between the numbers of users and the performance. A Pearson Coefficient of -0.18 indicates that there is no correlation.

After discussing reasons for clusters with unsatisfying performance and before discussing how this problem can be tackled in future work in Section VIII, we want introduce potential use cases for a recommender system as the presented one. Basically, we are considering two use cases:

1) Track Recommendation during Playlist Generation
2) Context-based Music Recommendations

Regarding the first use case, a possible implementation could be a recommender system supporting the user during the process of creating playlists. If a user creates a new playlists or adapts it, we can compute the distance of this playlist to the presented clusters and afterwards start the recommendation of tracks which fit into the playlist. The second use case is to recommender suitable music for various situations of the users throughout the whole day. I.e., it would be possible the recommended "get up" music in the morning and "driving" music on the way into the office and "concentration" music during the workday. However, in this case it would be necessary that the users specify their activities or rather the "type" of music they want to hear.

VII. Related Work

During the last decades (music) recommender systems have been from interest. Thus, a lot of research concerned with music recommendations is already present. There are basically three approaches if we categorize recommender systems regarding the input data or rather data sources they are using: *content-based (CB)* recommender systems, *context-based (CXB)* recommender systems and *collaborative filtering (CF)* based recommender systems [19]. Besides these, there are also various hybrid recommender systems, exploiting different recommendation techniques and data sources. A hybrid of the latter two recommender systems (CF and CXB) is presented in this work. In contradiction to the presented CF based approach, CB recommender systems are concerned with item features. In general, they use these features in order to find items, which are similar to a certain item i. Those items are the recommendation candidates for users interacted with item i. In case of music recommendations, these features (amongst others) are based on time and frequency characteristics of tracks [20]. Besides pure CB approaches, there are also several pure CF based approaches as presented in [1] and [2] and several hybrid approaches. [3] for instance combine a CF and CXB based approach. In their work, the geolocation is exploited as contextual information rather than playlist names as we do. There are also hybrid recommender combining CB and CF based approaches, i.e., the approach suggested by [21].

VIII. Conclusion and Future Work

In this section we first briefly summarize our main findings before we elaborate on future work in the second part.

We find that contextual information extracted from playlist names is a valuable resource for improving music recommender systems. We show that using our approach aggregating playlist names to contextual cluster and the incorporation of those clusters into the recommendation process boosts the performance of the recommender system by at least 33%. We further observe that not all contextual clusters boost the performance. This is the issue where we focus our future research on: Part of a future work will be to find an intelligent approach to determine which contextual clusters are valuable to music recommendations and which clusters are negligible. This would enable us a "fallback" to traditional CF or a CB method for certain contexts where we know that our approach is inferior. Another vague idea, related to "good" and "bad" clusters, is to further split contextual clusters using content-based features. The main idea behind is to tackle the problem of contextual clusters where the music preferences differ widely: CB features for instance can help us distinguish users preferring relaxing music from user preferring fast during work. Related to this, we are going to have a deeper look on the clusters itself and how we can interpret them, i.e., the genre distributions among the clusters might be interesting or the before-mentioned CB analysis if certain item based features appear frequently among certain contextual clusters.

References

[1] E. Zangerle, W. Gassler, and G. Specht, "Exploiting twitter's collective knowledge for music recommendations," in *Proc. of the 2nd Workshop on Making Sense of Microposts (#MSM2012): Big things come in small packages*, 2012.

[2] M. Pichl, Z. Eva, and G. Specht, "#nowplaying on #Spotify: Leveraging Spotify Information on Twitter for Artist Recommendations," in *Proc. of First Intl. Workshop in Mining the Social Web (Extended Proc. of the Intl. Conf. on Web Engineering (ICWE 2015)*, 2015, in press.

[3] M. Schedl and D. Schnitzer, "Hybrid Retrieval Approaches to Geospatial Music Recommendation," in *Proc. of the 35th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR 2013)*, 2013.

[4] M. Schedl, A. Vall, and K. Farrahi, "User Geospatial Context for Music Recommendation in Microblogs," in *Proc. of the 37th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR 2014)*, 2014.

[5] B.-j. Han, S. Rho, S. Jun, and E. Hwang, "Music emotion classification and context-based music recommendation," *Multimedia Tools and Applications*, vol. 47, no. 3, pp. 433–460, 2010.

[6] S. Rho, B.-j. Han, and E. Hwang, "Svr-based music mood classification and context-based music recommendation," in *Proc. of the 17th ACM Intl. Conf. on Multimedia*, 2009, pp. 713–716.

[7] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lke, and R. Schwaiger, "Incarmusic: Context-aware music recommendations in a car," in *E-Commerce and Web Technologies*, ser. Lecture Notes in Business Information Processing, C. Huemer and T. Setzer, Eds. Springer, 2011, vol. 85, pp. 89–100.

[8] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci, "Context relevance assessment and exploitation in mobile recommender systems," *Personal Ubiquitous Comput.*, vol. 16, no. 5, pp. 507–526, 2012.

[9] S. J. Cunningham, D. Bainbridge, and A. Falconer, "More of an art than a science: Supporting the creation of playlists and mixes." in *Proc. of 7th Intl. Conf. on Music Information Retrieval*, 2006, pp. 240–245.

[10] E. Zangerle, M. Pichl, W. Gassler, and G. Specht, "#nowplaying music dataset: Extracting listening behavior from twitter," in *Proc. of the 1st ACM Intl. Workshop on Internet-Scale Multimedia Management*, 2014, pp. 21–26.

[11] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[12] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[13] K. Mardia, J. Kent, and J. Bibby, *Multivariate analysis*, ser. Probability and mathematical statistics. London [u.a.]: Acad. Press, 1979.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[15] D. Brillinger, *Time Series: Data Analysis an Theory*, ser. Holden-Day Series in Time Series Analysis. Holden-Day, 1981.

[16] H. B. Mann and D. R. Whitney, "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other," *The Annals of Mathematical Statistics*, vol. 18, pp. 50–60, 1947.

[17] P. Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[18] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.

[19] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[20] M. Schedl, E. Gómez, and J. Urbano, "Music information retrieval: Recent developments and applications," *Foundations and Trends in Information Retrieval*, vol. 8, no. 2–3, pp. 127–261, 2014.

[21] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences," in *Proc. of the 7th Intl. Conf. on Music Information Retrieval (ISMIR 2006)*, 2006, pp. 296–301.