#nowplaying on #Spotify: Leveraging Spotify Information on Twitter for Artist Recommendations

Martin Pichl, Eva Zangerle, and Günther Specht

Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
{martin.pichl,eva.zangerle,guenther.specht}@uibk.ac.at

Abstract. The rise of the web enabled new distribution channels like online stores and streaming platforms, offering a vast amount of different products. For helping customers finding products according to their taste on those platforms, recommender systems play an important role. Besides focusing on the computation of the recommendations itself, in literature the problem of a lack of data appropriate for research is discussed. In order to overcome this problem, we present a music recommendation system exploiting a dataset containing listening histories of users, who posted what they are listening to at the moment on the microblogging platform Twitter. As this dataset is updated daily, we propose a genetic algorithm, which allows the recommender system to adopt its input parameters to the extended dataset. In the evaluation part of this work, we benchmark the presented recommender system against two baseline approaches. We show that the performance of our proposed recommender is promising and clearly outperforms the baseline.

Keywords: Music Recommender Systems, Collaborative Filtering, Social Media, Twitter

1 Introduction

The way how consumers access music has changed in recent years due to the rise of the web. Nowadays, consumers have the possibility to access a huge amount of different music using various devices and services. An example for such new services are music streaming platforms. The distribution and especially the inventory costs of such platforms are lower than the costs of traditional channels, i.e., brick and mortar stores. Due to this development, an increased amount of more diverse music is available, as additionally to bestsellers also niche music can be offered with low additional costs [1]. Besides commercial vendors like Spotify[1] or Pandora[2], there are also open platforms like SoundCloud[3] or Promo

[1] http://www.spotify.com
[2] http://www.pandora.com
[3] http://soundcloud.com

DJ[4], where users can upload and publish their own creations, increasing the diversity of available music. In contradiction to traditional channels like the radio, these new channels allow consumers to freely choose tracks they listen to and to create own playlists. One drawback of this freedom of choice is, that it is difficult for the customers to identify new tracks they like and want to listen to in this sheer amount of artists and tracks. As explained in Section 2, recommender systems can be seen as method to deal with information overload and a type of searching for information [7]. Thus, implementing a recommender system helps users discovering new tracks according to their taste, which increases user satisfaction.

As most data corpora in this field are owned by private companies, e.g., Spotify or Pandora mentioned above, this data is not or only publicly available limited. Thus, we propose to utilize Twitter and Spotify as new sources of publicly available data, which enables us to develop and evaluate our recommender system. Furthermore, we steadily enlarge the dataset. Since July 2011, the Research Group Databases and Information System of the University of Innsbruck is crawling tweets via the Twitter Streaming API. The whole process generating a dataset suitable for music recommendations and the basic statistics are presented in Section 3.1. In order to foster research in this field, the dataset is made available publicly[5].

Our main contribution, beside the dataset itself, is that it is possible to provide music recommendations by utilizing Twitter data. We propose an approach which adopts itself to this ever changing data. We show that a promising solution for dealing with these changes is a genetic logarithm (GA). Beside this, the evaluation of this recommender system shows, that in the case of CF, the track listening history seems to be more important than the artist listening history of a user.

In short, this paper addresses two problems: The first problem is the recommendation process itself. This problem is addressed by a collaborative filtering (CF) based recommender system, which is presented in more detail in Section 2. Afterwards, in Section 3, the second problem is addressed, which is a lack of publicly available and recent data appropriate for implementing and evaluating recommender system in academia [12]. This problem is addressed by exploiting Twitter and Spotify as a source of publicly available data. We briefly describe the generation of the dataset and the dataset itself before focusing on the evaluation in Section 3. Before we present the conclusions drawn from the evaluation in Section 5, related work is discussed in Section 4.

2 The Recommender System

In this section, the implementation of the artist recommender system is described in more detail. However, firstly we briefly explain the purpose of recommender systems and introduce the reader to the methods used.

[4] `http://promodj.com`
[5] available at: `http://dbis-twitterdata.uibk.ac.at/spotifyData/`

2.1 Background

In general, recommender systems can be seen as a method to deal with information overload [7] and a new type of searching for information. A common method for overcoming the information overload is to state one recommendation or a list of recommendations, which is much shorter than the initial list of all potential candidate items [7]. This is exactly what we do: Our recommender system helps users finding music they want to listen to or buy, by stating a short list of recommendations. In particular, our system provides users with artist recommendations, a user might be interested in.

In order to provide these artist recommendations, we rely on user-based collaborative filtering (CF). User-based CF recommends items by finding similar users, herein referred to as the nearest neighbours, and then suggesting items a user didn't interacted with, but his nearest neighbours interacted with. This is based on the assumption that like-minded users prefer the same items and that these user preferences remain stable over time [12]. For the music recommender system presented in this work, a user-item interaction states that a user listened to a certain track by a certain artist. Thus, we recommend artists the nearest neighbours of a user listened to but are new to the user himself.

A recommender system as ours, relies on certain input parameters, i.e., the number of nearest neighbours taken into consideration for retrieving the recommendations. In order to estimate suitable parameters, we implemented a genetic algorithm (GA) [10] as a possible solution. A GA is a heuristic for solving search and optimization problems [14]. The decision for performing the parameter optimization using GA was taken, as it would be computationally too expensive to compute the precision of our recommender system with all possible parameter combinations. The suitability of a GA for overcoming the problem of selecting and weighting input parameter, amongst others for recommender systems, was shown by Fong et al. [6]. The idea behind a GA, as introduced by Holland [10], is that the chromosomes of the fittest individuals of a population are inherited to the next generation. This inheritance, also called natural selection is achieved by recombining the genes of the fittest to new chromosomes by crossovers. Additionally, they are altered by random mutations and inversions [14].

How our artist recommender system computes the recommendations, on which input parameters it relies and how they are optimized is presented in the consecutive section.

2.2 Implementation

An overview of the recommendation process is shown in Figure 1. Using the Twitter Streaming API, a crawler is continually crawling for new tweets containing one of the following keywords: *nowplaying, listento* and *listeningto*. The update of the dataset, which means that the new tweets are resolved using the contained Spotify URL and stored in a database, can be done periodically. In our case this is done every 24 hours. This implies that our dataset is steadily enlarged and the optimal parameters may change. Thus, these parameters have to be updated along with the dataset. Therefore, an update of the dataset triggers the genetic algorithm (GA) which is updating the parameters for the music recommender system. This is why the actual recommendation process consists of two steps: In the first step, good input parameters have to be found, before providing actual recommendations using collaborative filtering (CF) in the second step.

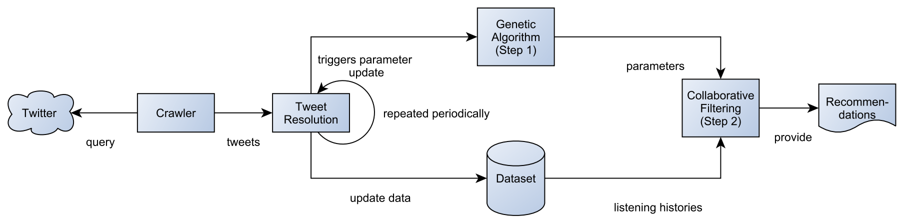

Fig. 1. The Recommendation Workflow

In the following, we describe our CF approach and present the input parameters the recommender system depends on. As already mentioned, User-based CF recommends items by finding similar users: the nearest neighbours. In our scenario, user listening histories are used for identifying the nearest neighbours and then computing the actual artist recommendations. As the artist listening history we consider the set of all artists a user i listened to ($artists_i$) in the training data and analogously, we consider the set of all tracks a user listened to as the track listening history ($tracks_i$). In order to compute the user similarities for determining the nearest neighbours, we applied the Jaccard Coefficient, as defined in Equation 1 [11], to the listing histories of two users i and j.

$$jaccard_{i,j} = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} \tag{1}$$

The Jaccard Coefficient measures the similarity of two sets S between 0 and 1. We compute this metric for the artist- as well as a track similarity as defined in Equations 2 and 3:

$$artistSim_{i,j} = \frac{|artists_i \cap artists_j|}{|artist_i \cup artists_j|} \tag{2}$$

$$trackSim_{i,j} = \frac{|tracks_i \cap tracks_j|}{|tracks_i \cup tracks_j|} \tag{3}$$

The final user similarity ($userSim$) is computed as the weighted average of both Jaccard Coefficients as depicted in Equation 4.

$$userSim = w_a * artistSim + w_t * trackSim. \tag{4}$$

The weights w_a and w_t determine the influence of the artist- and the track listening history on the user similarity, where $w_a + w_t = 1$. Thus, if $w_t = 0$, only the artist listening history is taken into consideration and analogously, if $w_a = 0$, only the track listening history is taken into consideration.

Using this user similarity, we are able to compute the similarity of a user i to all other users in the dataset and determine the k-nearest neighbours. These k-nearest neighbours are used to derive a set of possible recommendations. These recommendation candidates are items the nearest neighbours listened to, but have not been listened to by the user i.

For ranking the candidate items, we implemented two strategies: The first strategy (rs_1) computes a ranking coefficient r for every item i by counting how often an item occurs among the k-nearest neighbours, whereas the second strategy (rs_2) computes r by summing up the similarities of user i to any other user j found among the k-nearest neighbours K, who interacted with the same item as shown in Equation 5.

$$r_i = \sum_{j \in K} userSim_{i,j} \tag{5}$$

In order to perform the computation presented above, which is the second recommendation step, the following four parameters have to be set in the first step:

1. The weight of the $artistSim : w_a$
2. The weight of the $trackSim : w_s$
3. The number of nearest neighbours k
4. The ranking strategy R

These parameter form the genes of the GA. In order to build a initial population and allow mutations we generate random values for the float point and integer genes (w_a, w_t, k) and bit flips for the boolean gene (R). The *fitness* of an individual (in this case a recommender system) in the population is defined by the *precision* of this individual. We use the *precision* as the fitness measure, as we want to compute a short list of recommendations and to find all artists a user likes. For this so-called "find good items task", this is a reasonable metric [9]. Each individual has four genes forming the chromosome. The chromosome encodes the presented recommender parameters w_a, w_t, k and R. The *precision* of each individual is computed as described in Section 3.2.

These parameters should remain stable, as long as there are no changes in the dataset. Thus, the costly estimation of the parameters does not have to be performed prior each recommendation, but every time there is a change to the dataset, i.e., if it is updated as shown in Figure 1. After presenting the implementation of our recommender system, we present how the evaluation of this recommender system is conducted in the next section.

3 Recommendation Evaluation

In this section, the used dataset as well as the details of the recommender system evaluation are presented. As already mentioned, the evaluation is used to find suitable input parameters for the recommender system but also to show the performance of the recommender system.

3.1 Dataset

In contrast to other works aiming at extracting music information from Twitter, where the tweet's content is used to extract artist and track information from [19, 8, 15], we propose to exploit the subset of crawled tweets containing a URL leading to the website of the Spotify music streaming service. I.e., information about the artist and the track are extracted from the website mentioned in the tweet, rather than from the content of the tweet. This enables an unambiguous resolution of the tweets, in contradiction to the works mentioned above, where the text of the tweets is compared to entries in the reference database using some similarity measure. A typical tweet, published via Spotify, is depicted in the following: "#nowPlaying I Tried by Total on #Spotify http://t.co/ZaFH ZAokbV", where a user published that he or she listened to the track "I Tried" by the band "Total" on Spotify. Additionally, a shortened URL is provided. Besides this shortened URL, Twitter also provides the according resolved URL via its API. This allows for directly identifying all Spotify-URLs by searching for all URLs containing the string "spotify.com" or "spoti.fi". By following the identified URLs, the artist and the track can be extracted from the title tag of the according website.

Besides an unambiguous resolution and thus as clean dataset, utilizing Spotify tweets enables us to crawl additional information from Spotify about the track and about the user for future works. More details about this is presented in the outlook in Section 5.

The evaluation dataset is based on a snapshot of the dataset taken on April 30th, 2015 and is available online. This snapshot contains [513,489] unique $< user,\ artist,\ song >$-triples. Among the dataset we found 97,586 unique tracks by 40,593 artists, listened by 68,045 different users. One issue, which is addressed in a future work by user tweets crawling, is the data sparsity of the dataset. The sparsity can be seen in in Table 1, where the number of user with a minimum amount of tweets in the dataset is stated. Data sparsity is an issue, as normally the performance of CF based recommender systems increases with the detailedness of a user profile. This problem of the data sparsity will be addressed in a future work, as described in Section 5.

3.2 Evaluation Setup

In order to assess the performance of the recommender system, we conducted an offline evaluation. The details of this evaluation are presented in this section.

Table 1. Number of Tweets and Number of Users

Number of Tweets	Number of Users
> 0	68,045
> 1	38,906
> 10	9,111
> 100	516
$> 1{,}000$	12

For computing the *precision* as depicted in Equation 6, we split the listening history of each user into a training- and a test set. The test set was created by randomly removing one third of the listening events. The other two thirds of the listening events were actually used to compute the recommendations. If a computed recommendation was found in the test set, it was considered as a true positive.

$$precison = \frac{True\ Positives}{Number\ of\ Recommendations} \tag{6}$$

Furthermore we computed the *recall* as shown in Equation 7. For computing the *recall* metric, all items in the test set are considered as relevant items (and hence are desirable to recommend to the user). The recall metric describes the fraction of relevant artists who are recommended., i.e., when recommending 5 items, even if all items are considered relevant, the maximum recall is still only 50% if the size of the test set is 10. Thus, in this evaluation setup, *recall* is bound by an upper limit, which is the number of recommended items divided by the size of the test set.

$$recall = \frac{True\ Positives}{Size\ of\ the\ Testset} \tag{7}$$

Finally, for each user, we computed the *precision* and *recall* metrics for recommending a different number of artists depending on the size of the test set as depicted in Equation 8. We varied the percentage value p between 10% up to 100% and repeated each experiment 10 times in order to estimate the variance.

$$recommendations = p * Size\ of\ the\ Testset \tag{8}$$

3.3 Detection of Optimal Parameters

As stated in Section 2, for finding good input parameters a GA [10] is implemented. To measure the fitness of a population, we used the *precision* as argued in Section 2.2. The size of the population was set to 10 individuals. As depicted in Figure 2, with this configuration and using a snapshot of the dataset taken on April 30th, 2015, a good solution is found after 6 iterations. On average, a solution was found after 4.14 iterations ($SD = 2.27$), thus we propose to terminate the recommender after 7 iterations ($M + SD = 4.14 + 2.27 = 6.41$).

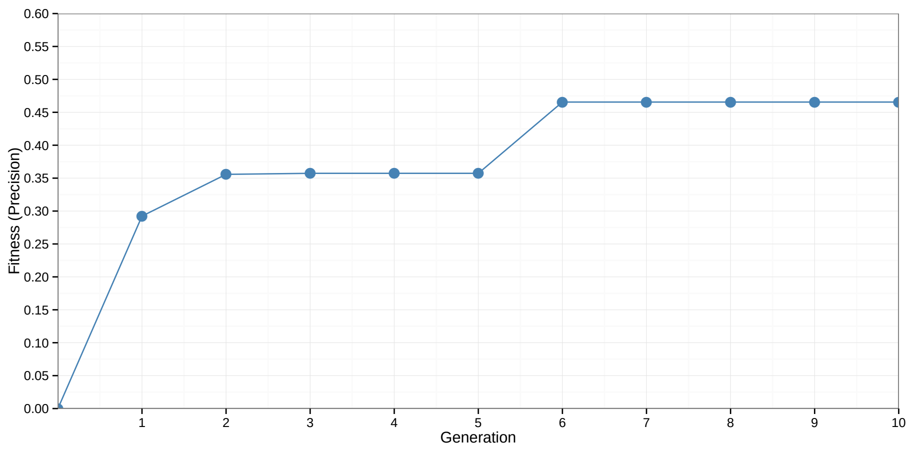

Fig. 2. Fitness (Precision) of the best Individual in each Generation

This solution, a recommender system with the parameters as stated below, is used for the performance evaluation of the recommender system in the following section.

1. The weight of the $artistSim : w_a = 0.21$
2. The weight of the $trackSim : w_s = 0.94$
3. The number of nearest neighbours $k = 59$
4. The ranking strategy $R = rs_2$

From the result we can see that for the current snapshot of the dataset, the track listening history seems to be more significant for calculating the user similarity. Furthermore, we can also see that there was a mutation as $w_a + w_t \neq 1$.

3.4 Performance of the Recommender System

The parameters detected in Section 3.3 were used in order to perform the performance analysis of our proposed recommender system (new Approach). We compare the performance of our recommender system to two baseline recommendation approaches: Baseline approach number 1 (BA1) simply recommends the most popular items in the dataset to each of the users, whereas baseline approach number 2 (BA2) is a standard user-based CF approach without any further optimization than the number of nearest neighbours. Thus, the parameters of the second baseline recommendation approach are: $w_a = 1,\ w_t = 0,\ k = 59$. The performance of all three recommender systems is shown in Figure 3 and Table 2 (standard deviations in parentheses).

Although our proposed recommender system clearly outperforms both baseline approaches and thus is a promising approach, the *precision* drops rather fast with the number of recommendations (see Figure 3). This is, as the chance

Table 2. Precision and Recall of the Recommender Systems

	BA1		**BA2**		**new Approach**	
p	Precision	Recall	Precision	Recall	Precision	Recall
0.1	0.156 (0.039)	0.024 (0.003)	0.316 (0.024)	0.015 (0.001)	0.466 (0.023)	0.022 (0.001)
0.5	0.117 (0.002)	0.026 (0.001)	0.117 (0.006)	0.060 (0.003)	0.253 (0.006)	0.120 (0.003)
1.0	0.081 (0.007)	0.032 (0.002)	0.096 (0.003)	0.081 (0.003)	0.181 (0.005)	0.158 (0.005)

of false positives increases with the number of recommendations, if the size of the test set is kept constant. However, as discussed in Section 5, we already experimenting with further performance improvements by incorporating the user's context into the recommendation process.

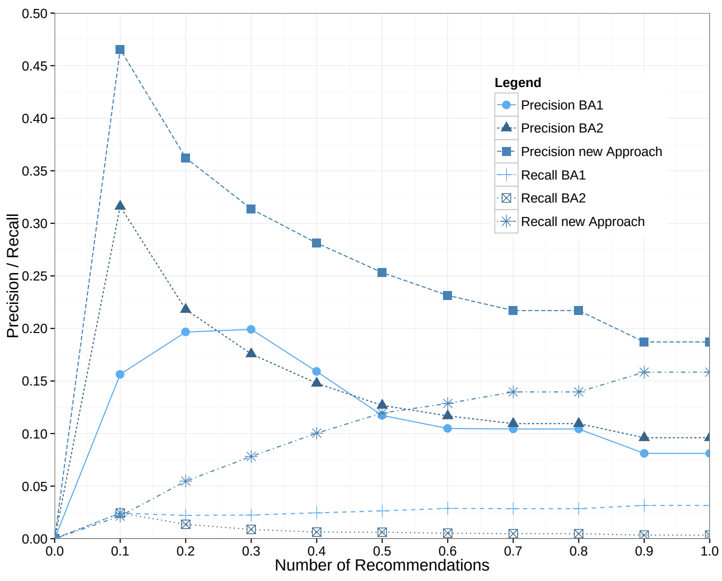

Fig. 3. Precision and Recall of the Recommender Systems

4 Related Work

Work related to the approach presented in this paper is concerned with both problem domains addressed in this work: (i) the generation of appropriate datasets for research in the field of recommender systems and (ii) research in the field of (music) recommender systems itself.

Regarding (i), Hauger et al. [8] published the Million Musical Tweets Dataset (MMTD). The MMTD is a static dataset containing about one million music related tweets. All tweets in this dataset are tagged with a geolocation. Similar to ours, their dataset also suffers from a high data sparsity or rather a low number of tweets per user. Another very popular dataset, containing one million listening events, is based on data logged by the music discovery service Last.fm. The Million Song Dataset (MSD) [2], released 2011, contains information about one million songs from different sources. Beside real user play counts, it provides audio features of the songs and is therefore suitable for CF, content-based (CB) and hybrid recommender systems. Besides those comparable large datasets, Yahoo! published numerous datasets[6] containing ratings for artists and songs suitable for CF. The largest Yahoo! dataset contains 136,000 songs along with ratings given by 1.8 million users. The data itself was gathered by monitoring users using the Yahoo! Music Services between 2002 and 2006 and thus, the dataset is less recent. Additionally to the ratings, the Yahoo dataset contains genre information which can be exploited by a hybrid recommender system. Celma, based on his research [4], also provides a music dataset, containing data retrieved from last.fm. It is comparably small and contains user, artists and play counts as well as the MusicBrainz identifiers for 360,000 users. This dataset was published in 2010 [4]. Another interesting dataset, which is in contradiction to ours suitable for recommending movies, is the MovieTweetings dataset[7] published by Dooms et al. [5]. As the dataset used in this work, it is continually updated and contains movies and the corresponding user ratings extracted from structured tweets tweeted via the Internet Movie Database (IMDb)[8].

Beside works concerned with generating datasets, related work is also concerned with recommendation approaches in general. There are various approaches for implementing recommender systems. Burke [3] classifies recommender systems by the data or rather the data source they use: CB recommendation approaches use information about items, whereas recommender systems based on CF take information about all users of a system or rather the user's taste into account. Demographic-based recommender systems use demographic data and Knowledge-based recommender systems rely on user needs. Finally, there are Hybrid recommender systems, combining the different approaches mentioned above. In the field of music recommendation, different hybrid approaches have been evaluated. Yoshii et al. state, that hybrid recommender system exploiting CF and CB data can outperform recommender systems based solely on CF [18] and Schedl et al. [16, 17] argue, that a CF based recommender systems incorporating the geolocation works best.

Also GAs have been used for recommender systems in similar settings than ours. Hyun-Tae Kim et al. [13] propose a CB recommender system based on different audio features. They use an interactive genetic algorithm (IGA), where the users are evaluating the fitness of each population manually, in order to enable the system the music preferences of the users and recommend suitable music tracks. Instead of relying on a dataset, they conduct a real user experiment. They claim that their approach is a success, however the conducted experiment

[6] available at: `http://webscope.sandbox.yahoo.com/catalog.php?datatype=r`
[7] `http://github.com/sidooms/MovieTweetings`
[8] `http://www.imdb.com`

incorporates only ten users. Similar to us, Fong et al. [6] address the problem of finding input parameters using a GA. The use a GA for selecting and weighting different features of their CB recommender system suitable for recommending movies.

5 Discussion and Future Work

The recommender system implemented in this work delivers promising results compared to the baseline approaches. From evaluations results stated in Table 2, we can derive that for the current snapshot of the dataset, the track listening history seems to be more significant for calculating the user similarity. We lead this back to the fact, that the track implicitly contains the artist. Besides this we see that at the current stage, the performance of the proposed recommender system is limited if the number of recommendations is high. In order to further boost the performance, we will aim at (i) enlarging the dataset by user playlist crawling on Spotify to reduce the data sparsity and (ii) incorporating the user's listening context. The idea behind (ii) is, that people are listening to different music in different situations. First experiments have shown, that incorporating the context is a promising approach: We already crawling additional information like favourite tracks and playlist names. From those playlist names, we are able to extract context-based information: We observed for instance playlists for certain events like Christmas, weddings or workouts but also playlist of certain moods, i.e., chill, relax, in love or the like. At the moment we are experimenting how to integrate this information into the presented recommender system.

6 Conclusion

Our proposed recommender system uses a genetic algorithm for optimizing the input parameters to the presented, steadily updated dataset. For computing the actual recommendations, it uses user-based collaborative filtering with a hybrid user-similarity computation. Although the implemented algorithm is simple, the performance of the recommender systems is promising and clearly outperforms the baseline approaches, namely recommending the most popular artists and standard user-based collaborative filtering. Nevertheless, as discussed in the previous section, there is space left for further performance improvements. Thus, we integrate context-based information in the next step in order to design a more specialized and more user-centric algorithm.

References

1. C. Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More.* Hyperion, 2006.
2. T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pages 591–596. University of Miami, 2011.

3. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
4. Ò. Celma. *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space.* Springer, 2010.
5. S. Dooms, T. De Pessemier, and L. Martens. Movietweetings: a movie rating dataset collected from twitter. In *Workshop on Crowdsourcing and Human Computation for Recommender Systems at the 7th ACM Conference on Recommender Systems (RecSys 2013)*, 2013.
6. S. Fong, Y. Ho, and Y. Hang. Using genetic algorithm for hybrid modes of collaborative filtering in online recommenders. In *Proceedings of the Eighth International Conference on Hybrid Intelligent Systems (HIS 2008)*, pages 174–179, 2008.
7. D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.
8. D. Hauger, M. Schedl, A. Kosir, and M. Tkalcic. The million musical tweet dataset - what we can learn from microblogs. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 189–194, 2013.
9. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.
10. J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, USA, 1975.
11. P. Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, 1912.
12. D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction.* Cambridge University Press, 2010.
13. H.-T. Kim, E. Kim, J.-H. Lee, and C. W. Ahn. A recommender system based on genetic algorithm for music data. In *Proceedings of the 2nd International Conference on Computer Engineering and Technology (ICCET 2010)*, volume 6, pages V6–414–V6–417, 2010.
14. M. Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, MA, USA, 1998.
15. M. Schedl and D. Schnitzer. Hybrid Retrieval Approaches to Geospatial Music Recommendation. In *Proceedings of the 35th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2013)*, 2013.
16. M. Schedl and D. Schnitzer. Location-Aware Music Artist Recommendation. In *Proceedings of the 20th International Conference on MultiMedia Modeling (MMM 2014)*, 2014.
17. M. Schedl, A. Vall, and K. Farrahi. User Geospatial Context for Music Recommendation in Microblogs. In *Proceedings of the 37th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2014)*, 2014.
18. 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 *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 296–301, 2006.
19. E. Zangerle, W. Gassler, and G. Specht. Exploiting twitter's collective knowledge for music recommendations. In *Proceedings of the 2nd Workshop on Making Sense of Microposts (#MSM2012)*, pages 14–17, 2012.