# Combining Spotify and Twitter Data for Generating a Recent and Public Dataset for Music Recommendation

Martin Pichl
Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
martin.pichl@uibk.ac.at

Eva Zangerle
Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
eva.zangerle@uibk.ac.at

Günther Specht
Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
guenther.specht@uibk.ac.at

## ABSTRACT

In this paper, we present a dataset based on publicly available information. It contains listening histories of Spotify users, who posted what they are listening at the moment on the micro blogging platform Twitter. The dataset was derived using the Twitter Streaming API and is updated regularly. To show an application of this dataset, we implement and evaluate a pure collaborative filtering based recommender system. The performance of this system can be seen as a baseline approach for evaluating further, more sophisticated recommendation approaches. These approaches will be implemented and benchmarked against our baseline approach in future works.

## Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.2.8 [**Database Applications**]: Data mining

## General Terms

Algorithms, Experimentation

## Keywords

Music Recommender Systems, Collaborative Filtering, Social Media

## 1. INTRODUCTION

More and more music is available to be consumed, due to new distribution channels enabled by the rise of the web. Those new distribution channels, for instance music streaming platforms, generate and store valuable data about users and their listening behavior. However, most of the time the data gathered by these companies is not publicly available. There are datasets available based on such private data corpora, which are widely used for implementing and evaluating recommender systems, i.e., the million song dataset (MSD) [4], however such datasets like the MSD often are not recent anymore. Thus, in order to address the problem of a lack of recent and public available data for the development and evaluation of recommender systems, we exploit the fact that many users of music streaming platforms post what they are listening to on the microblogging Twitter. An example for such a tweet is "#NowPlaying Human (The Killers) #craigcardiff #spotify http://t.co/N08f2MsdSt". Using a dataset derived from such tweets, we implement and evaluate a collaborative filtering (CF) based music recommender system and show that this is a promising approach. Music recommender systems are of interest, as the volume and variety of available music increased dramatically, as mentioned in the beginning. Besides commercial vendors like Spotify[1], there are also open platforms like SoundCloud[2] or Promo DJ[3], which foster this development. On those platforms, users can upload and publish their own creations. As more and more music is available to be consumed, it gets difficult for the user or rather customer to navigate through it. By giving music recommendations, recommender systems help the user to identify music he or she wants to listen to without browsing through the whole collection. By supporting the user finding items he or she likes, the platform operators benefit from an increased usability and thus increase the customer satisfaction.

As the recommender system implemented in this work delivers suitable results, we will gradually enlarge the dataset by further sources and assess how the enlargements influences the performance of the recommender system in future work. Additionally, as the dataset also contains time stamps and a part of the captured tweets contains a geolocation, more sophisticated recommendation approaches utilizing these additional context based information can be compared against the baseline pure CF-based approach in future works.

The remainder of this paper is structured as follows: in Section 2 we present the dataset creation process as well as the dataset itself in more detail. Afterwards, in Section 3 we briefly present the recommendation approach, which is evaluated in Section 4. Before we present the conclusion drawn from the evaluation on Section 6, related work is discussed in Section 5.



In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the $26^{th}$ GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

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

## 2. THE SPOTIFY DATASET

In this Section, the used dataset [4] for developing and evaluating the recommender system is presented.

### 2.1 Dataset Creation

For the crawling of a sufficiently large dataset, we relied on the Twitter Streaming API which allows for crawling tweets containing specified keywords. Since July 2011, we crawled for tweets containing the keywords *nowplaying, listento* and *listeningto*. Until October 2014, we were able to crawl more than 90 million tweets. In contrast to other contributions aiming at extracting music information from Twitter, where the tweet's content is used to extract artist and track information from [17, 7, 16], we propose to exploit the subset of crawled tweets containing a URL leading to the website of the Spotify music streaming service[5]. 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 us an unambiguous resolution of the tweets, in contradiction to the contributions 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. For instance, the title of the website behind the URL stated above is "<title>I tried by Total on Spotify </title>". Using the regular expression "<title>(.*) by (.*) on.*</title>" the name of the track (group 1) and the artist (group 2) can be extracted.

By applying the presented approach to the crawled tweets, we were able to extract artist and track information from 7.08% of all tweets or rather 49.45% of all tweets containing at least one URL. We refer to the subset of tweets, for which we are able to extract the artist and the track, as "matched tweets". An overview of the captured tweets is given in Table 1. 1.94% of the tweets containing a Spotify-URL couldn't be matched due to HTTP 404 Not Found and HTTP 500 Internal Server errors.

| Restriction | Number of Tweets | Percentage |
|---|---|---|
| None | 90,642,123 | 100.00% |
| At least one URL | 12,971,482 | 14.31% |
| A Spotify-URL | 6,541,595 | 7.22% |
| Matched | 6,414,702 | 7.08% |

**Table 1: Captured and Matched Tweets**

Facilitating the dataset creation approach previously presented, we are able to gather 6,414,702 tweets and extract artist and track data from the contained Spotify-URLs.

[4]available at: `http://dbis-twitterdata.uibk.ac.at/spotifyDataset/`

[5]`http://www.spotify.com`

### 2.2 Dataset Description

Based on the raw data presented in the previous Section, we generate a final dataset of <user, artist, track>-triples which contains 5,504,496 tweets of 569,722 unique users who listened to 322,647 tracks by 69,271 artists. In this final dataset, users considered as not valuable for recommendations, i.e., the @SpotifyNowPlay Twitter account which retweets tweets sent via @Spotify, are removed. These users were identified manually by the authors.

As typical for social media datasets, our dataset has a long-tailed distribution among the users and their respective number of posted tweets [5]. This means that there are only a few number of users tweeting rather often in this dataset and numerous users are tweeted rarely which can be found in the long-tail. This long-tailed distribution can be seen in Table 2 and Figure 1, where the logarithm of the number of tweets is plotted against the corresponding number of users.

| Number of Tweets | Number of Users |
|---|---|
| >0 | 569,722 |
| >1 | 354,969 |
| >10 | 91,217 |
| >100 | 7,419 |
| >1,000 | 198 |

**Table 2: Number of Tweets and Number of Users**

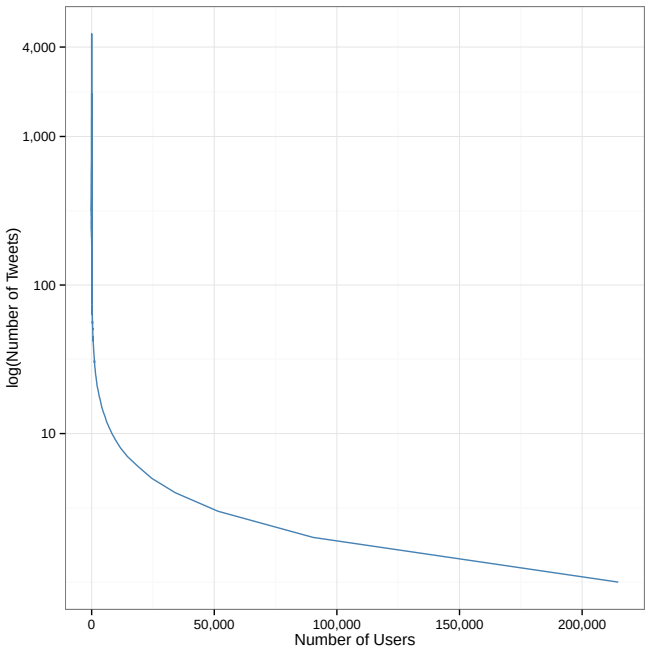


**Figure 1: Number of Tweets versus Number of Users**

The performance of a pure collaborative filtering-based recommender system increases with the detailedness of a user profile. Especially for new users in a system, where no or only little data is available about them, this poses a problem as no suitable recommendations can be computed. In our case, problematic users are users who tweeted rarely and thus can be found in the long tail.

Besides the long-tail among the number of posted tweets, there is another long-tail among the distribution of the artist play-counts in the dataset: there are a few popular artists occurring in a large number of tweets and many artists are mentioned only in a limited number of tweets. This is shown in Figure 2, where the logarithm of the number of tweets in which an artist occurs in (the play-count) is plotted against the number of artists. Thus, this plot states how many artists are mentioned how often in the dataset.

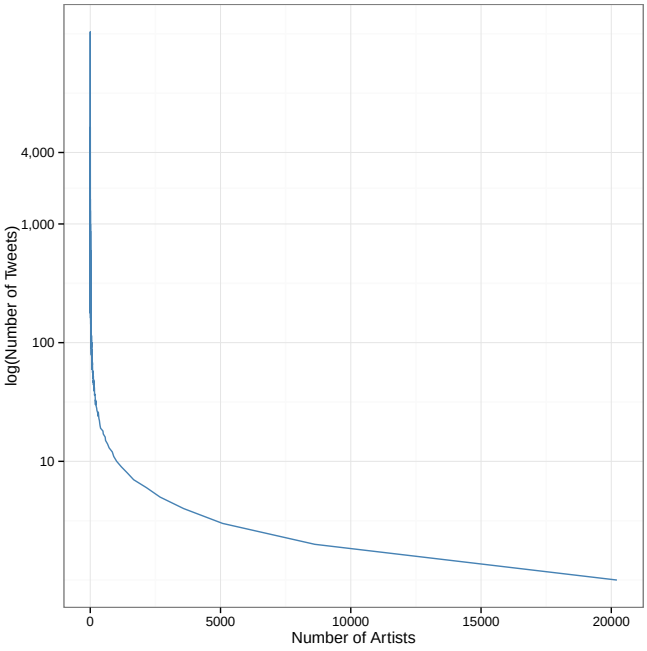


**Figure 2: Play-Count versus Number of Artists**

How the presented dataset is used as input- and evaluation data for a music recommender system, is presented in the next Section.

# 3. THE BASELINE RECOMMENDATION APPROACH

In order to present how the dataset can be applied, we use our dataset as input and evaluation data for an artist recommendation system. This recommender system is based on the open source machine learning library Mahout[2]. The performance of this recommender system is shown in Section 4 and serves as a benchmark for future work.

## 3.1 Recommendation Approach

For showing the usefulness of our dataset, we implemented a User-based CF approach. User-based CF recommends items by solely utilizing past user-item interactions. For the music recommender system, a user-item interaction states that a user listened to a certain track by a certain artist. Thus, the past user-item interactions represent the listening history of a user. In the following, we describe our basic approach taken for computing artist recommendations and provide details about the implementation.

In order to estimate the similarity of two users, we computed a linear combination of the Jaccard-Coefficients [10] based on the listening histories of the user. The Jaccard-Coefficient is defined in Equation 1 and measures the proportion of common items in two sets.

$$jaccard_{i,j} = \frac{|A_i \cap A_j|}{|A_i \cup A_j|} \tag{1}$$

For each user, there are two listening histories we take into consideration: the set of all tracks a user listened to and the set of all artists a user listened to. Thus, we are able to compute a *artist similartiy (artistSim)* and a *track similarity (trackSim)* as shown in Equations 2 and 3.

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

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

The final user similarity is computed using a weighted average of both, the *artistSim* and *trackSim* as depicted in Equation 4.

$$sim_{i,j} = w_a * artistSim_{i,j} + w_t * trackSim_{i,j} \tag{4}$$

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. We call such a recommender system an artist-based recommender system. Analogously, if $w_a = 0$ we call such a recommender system track-based. If $w_a > 0 \wedge w_t > 0$, both the artist- and track listening histories are used. Hence, we facilitate a hybrid recommender system for artist recommendations.

The presented weights have to be predetermined. In this work, we use a grid-search for finding suitable input parameter for our recommender system as described in Section 4.2.

# 4. EVALUATION

In this Section we present the performance of the implemented artist recommender system, but also discuss the limitations of the conducted offline evaluation.

## 4.1 Evaluation Setup

The performance of the recommender system with different input parameters was evaluated using *precision* and *recall*. Although we focus on the *precision*, for the sake of completeness we also include the *recall* into the evaluation, as this is usual in the field of information retrieval [3]. The metrics were computed using a Leave-$n$-Out algorithm, which can be described as follows:

1. Randomly remove $n$ items from the listening history of a user
2. Recommend $m$ items to the user
3. Calculate *precision* and *recall* by comparing the $m$ recommended and the $n$ removed items
4. Repeat step 1 to 3 $p$ times
5. Calculate the mean *precision* and the mean *recall*

Each evaluation in the following Sections has been repeated five times ($p = 5$) and the size of the test set was fixed to 10 items ($r = 10$). Thus, we can evaluate the performance of the recommender for recommending up to 10 items.

## 4.2 Determining the Input Parameters

In order to determine good input parameters for the recommender system, a grid search was conducted. Therefore, we define a grid of parameters and the possible combinations are evaluated using a performance measure [9]. In our case, we relied on the *precision* of the recommender system (cf. Figure 3), as the task of a music recommender system is to find a certain number of items a user will listen to (or buy), but not necessarily to find all good items. *Precision* is a reasonable metric for this so called *Find Good Items* task [8] and was assessed using the explained Leave-$n$-Out algorithm. For this grid search, we recommended one item and the size of the test set was fixed to 10 items. In order to find good input parameters, the following grid parameters determining the computation of the user similarity were altered:

- Number of nearest neighbors $k$
- Weight of the artist similarity $w_a$
- Weight of the track similarity $w_t$

The result can be seen in Figure 3. For our dataset it holds, that the best results are achieved with a track-based recommender system ($w_a = 0$,$w_t = 1$) and 80 nearest neighbors ($k = 80$). Thus, for the performance evaluation of the recommender system in the next Section, we use the following parameters:

- Number of nearest neighbors 80
- Weight of the artist similarity 0
- Weight of the track similarity 1

## 4.3 Performance of the Baseline Recommender System

In this Section, the performance of the recommender system using the optimized input parameters is presented. Prior to the evaluation, we also examined real implementations of music recommender systems: Last.fm, a music discovery service, for instance recommends 6 artists[6] when displaying a certain artist. If an artist is displayed on Spotify[7], 7 similar artists are recommended at the first page. This number of items also corresponds to the work of Miller [11], who argues that people are able to process about 7 items at a glance, or rather that the span of attention is too short for processing long lists of items. The *precision@6* and the *precision@7* of our recommender are 0.20 and 0.19, respectively. In such a setting, 20% of the recommended items computed by the proposed recommender system would be a hit. In other words, a customer should be interested in at least in two of the recommended artists. An overview about the *precision@n* of the recommender is given in Table 3.

[6] `http://www.last.fm/music/Lana+Del+Rey`
[7] `http://play.spotify.com/artist/00FQb4jTyendYWaN8pK0wa`

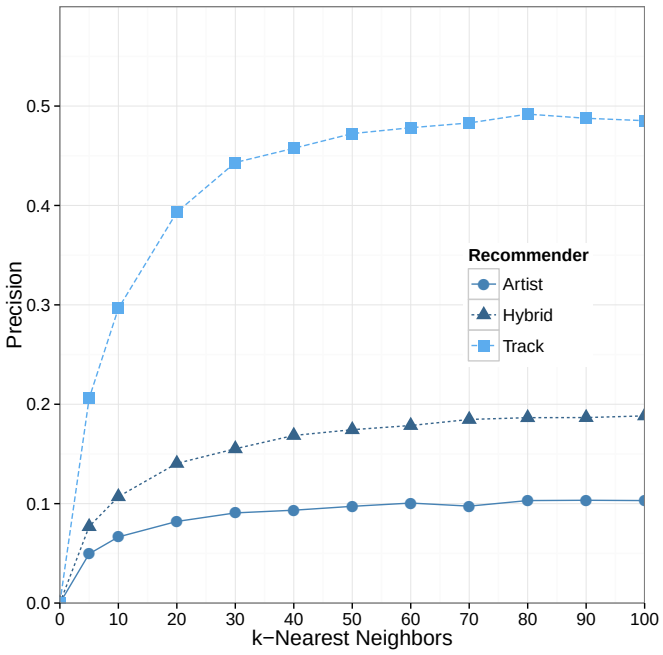


**Figure 3: Precision and Recall of the Track-Based Recommender**

| n | Precision | Recall | Upper Bound |
|---|---|---|---|
| 1 | 0.49 | 0.05 | 0.10 |
| 5 | 0.23 | 0.11 | 0.50 |
| 6 | 0.20 | 0.12 | 0.60 |
| 7 | 0.19 | 0.13 | 0.70 |
| 10 | 0.15 | 0.15 | 1.00 |

**Table 3: Precision and Recall of the Track-Based Recommender**

As shown in Figure 4, with an increasing number of recommendations, the performance of the presented recommender system declines. Thus, for a high number of recommendations the recommender system is rather limited. This is, as the chance of false positives increases if the size of the test set is kept constant. For computing the *recall* metric, the 10 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% as 10 items are considered as relevant. Thus, in the evaluation setup, recall is bound by an upper limit, which is the number of recommended items divided by the size of the test set.

## 4.4 Limitations of the Evaluation

Beside discussing the results, it is worth to mention also two limitations in the evaluation approach: First, only recommendations for items the user already interacted with can be evaluated [5]. If something new is recommended, it can't be stated whether the user likes the item or not. We can only state that it is not part of the user's listening history in our dataset. Thus, this evaluation doesn't fit to the per-

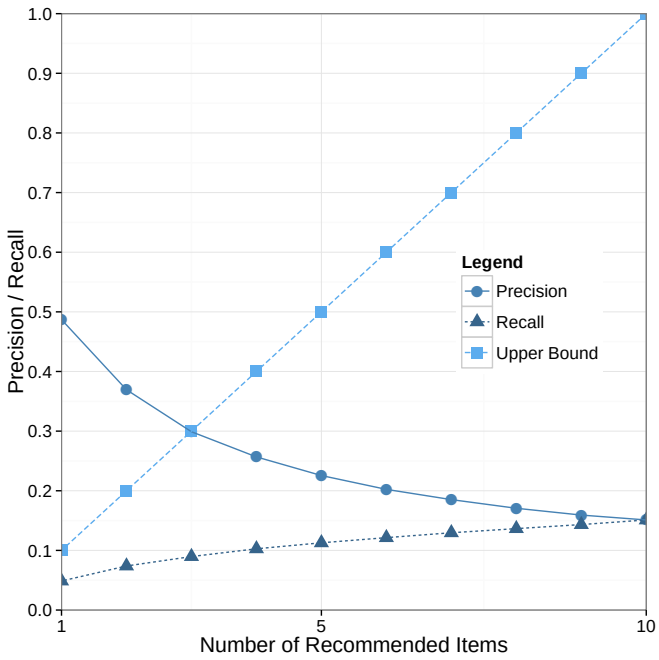


**Figure 4: Precision and Recall of the Track-Based Recommender**

fectly to the intended use of providing recommendations for new artists. However, this evaluation approach enabled us to find the optimal input parameters using a grid search. Secondly, as we don't have any preference values, the assumption that a certain user likes the artist he/she listened to, has to be made.

Both drawbacks can be eliminated by conducting a user-centric evaluation [5]. Thus, in a future work, it would be worth to conduct a user-experiment using the optimized recommender system.

## 5. RELATED WORK

As already mentioned in the introduction, there exist several other publicly available datasets suitable for music recommendations. A quick overview of these datasets is given in this Section.

One of the biggest available music datasets is the Million Song Dataset (MSD) [4]. This dataset 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-, CB- and hybrid recommender systems. At the moment, the Taste Profile subset[8] of the MSD is bigger than the dataset presented in this work, however it was released 2011 and is therefore not as recent.

Beside the MSD, also Yahoo! published big datasets[9] containing ratings for artists and songs suitable for CF. The biggest dataset contains 136,000 songs along with ratings given by 1.8 million users. Additionally, the genre information is provided in the dataset. The data itself was gathered by monitoring users using the Yahoo! Music Services between 2002 and 2006. Again, the MSD dataset, the Yahoo dataset is less recent. Additionally to the ratings, the Yahoo dataset contains genre information which can be exploited by a hybrid recommender system.

Celma also provides a music dataset, containing data retrieved from last.fm[10], a music discovery service. It contains user, artists and play counts as well as the MusicBrainz identifiers for 360,000 users. This dataset was published in 2010 [5].

Beside the datasets presented above, which are based on data of private companies, there exist several datasets based on publicly available information. Sources exploited have been websites in general [12, 15, 14], Internet radios posting their play lists [1] and micro-blogging platforms, in particular Twitter [17, 13]. However, using these sources has a drawback: For cleaning and matching the data, high effort is necessary.

One of the most similar datasets to the dataset used in this work, is the Million Musical Tweets Dataset [11] dataset by Hauger et al. [7]. Like our dataset, it was created using the Twitter streaming API from September 2011 to April 2013, however, all tweets not containing a geolocation were removed and thus it is much smaller. The dataset contains 1,086,808 tweets by 215,375 users. Among the dataset, 25,060 unique artists have been identified [7].

Another dataset based on publicly available data which is similar to the MovieLens dataset, is the MovieTweetings dataset published by Dooms et al. [6]. The MovieTweetings dataset is continually updated and has the same format as the MovieLens dataset, in order to foster exchange. At the moment, a snapshot containing 200,000 ratings is available[12]. The dataset is generated by crawling well-structured tweets and extracting the desired information using regular expressions. Using this regular expressions, the name of the movie, the rating and the corresponding user is extracted. The data is afterwards linked to the IMDb, the Internet Movie Database [13].

## 6. CONCLUSION AND FUTURE WORK

In this work we have shown that the presented dataset is valuable for evaluating and benchmarking different approaches for music recommendation. We implemented a working music recommender systems, however as shown in Section 4, for a high number of recommendations the performance of our baseline recommendation approach is limited. Thus, we see a need for action at two points: First we will enrich the dataset with further context based information that is available: in this case this can be the time stamp or the geolocation. Secondly, hybrid recommender system utilizing this additional context based information are from interest. Therefore, in future works, we will focus on the implementation of such recommender systems and compare them to the presented baseline approach. First experiments were already conducted with a recommender system trying to exploit the geolocation. Two different implementations are evaluated at the moment: The first uses the normalized linear distance between two users for approximating a user

[8] http://labrosa.ee.columbia.edu/millionsong/tasteprofile
[9] available at: http://webscope.sandbox.yahoo.com/catalog.php?datatype=r
[10] http://www.last.fm
[11] available at: http://www.cp.jku.at/datasets/MMTD/
[12] https://github.com/sidooms/MovieTweetings
[13] http://www.imdb.com

similarity. The second one, which in an early stage of evaluation seems to be the more promising one, increases the user similarity if a certain distance threshold is underrun. However, there remains the open question how to determine this distance threshold.

## 7. REFERENCES


[1] N. Aizenberg, Y. Koren, and O. Somekh. Build your own music recommender by modeling internet radio streams. In *Proceedings of the 21st International Conference on World Wide Web (WWW 2012)*, pages 1–10. ACM, 2012.

[2] Apache Software Foundation. What is Apache Mahout?, March 2014. Retrieved July 13, 2014, from http://mahout.apache.org.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology behind Search (2nd Edition) (ACM Press Books)*. Addison-Wesley Professional, 2 edition, 2011.

[4] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In A. Klapuri and C. Leider, editors, *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pages 591–596. University of Miami, 2011.

[5] Ò. Celma. *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer, 2010.

[6] 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.

[7] D. Hauger, M. Schedl, A. Kosir, and M. Tkalcic. The million musical tweet dataset - what we can learn from microblogs. In A. de Souza Britto Jr., F. Gouyon, and S. Dixon, editors, *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 189–194, 2013.

[8] 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, Jan. 2004.

[9] C. W. Hsu, C. C. Chang, and C. J. Lin. *A practical guide to support vector classification*. Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2003.

[10] P. Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, Feb. 1912.

[11] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. 62:81–97, 1956.

[12] A. Passant. dbrec - Music Recommendations Using DBpedia. In *Proceedings of the 9th International Semantic Web Conference (ISWC 2010)*, volume 6497 of *Lecture Notes in Computer Science*, pages 209–224. Springer Berlin Heidelberg, 2010.

[13] M. Schedl. Leveraging Microblogs for Spatiotemporal Music Information Retrieval. In *Proceedings of the 35th European Conference on Information Retrieval (ECIR 2013)*, pages 796 – 799, 2013.

[14] M. Schedl, P. Knees, and G. Widmer. Investigating web-based approaches to revealing prototypical music artists in genre taxonomies. In *Proceedings of the 1st International Conference on Digital Information Management (ICDIM 2006)*, pages 519–524. IEEE, 2006.

[15] M. Schedl, C. C. Liem, G. Peeters, and N. Orio. A Professionally Annotated and Enriched Multimodal Data Set on Popular Music. In *Proceedings of the 4th ACM Multimedia Systems Conference (MMSys 2013)*, pages 78–83, February–March 2013.

[16] 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.

[17] 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.