MUSE: A MUSIC RECOMMENDATION MANAGEMENT SYSTEM

Martin Przyjaciel-Zablocki, Thomas Hornung, Alexander Schätzle,
Sven Gauß, Io Taxidou, Georg Lausen
Department of Computer Science, University of Freiburg
zablocki,hornungt,schaetzle,gausss,taxidou,lausen@informatik.uni-freiburg.de

ABSTRACT

Evaluating music recommender systems is a highly repetitive, yet non-trivial, task. But it has the advantage over other domains that recommended songs can be evaluated immediately by just listening to them.

In this paper, we present MUSE – a music recommendation management system – for solving the typical tasks of an in vivo evaluation. MUSE provides the typical off-the-shelf evaluation algorithms, offers an online evaluation system with automatic reporting, and by integrating online streaming services also a legal possibility to evaluate the quality of recommended songs in real time. Finally, it has a built-in user management system that conforms with state-of-the-art privacy standards. New recommender algorithms can be plugged in comfortably and evaluations can be configured and managed online.

1. INTRODUCTION

One of the hallmarks of a good recommender system is a thorough and significant evaluation of the proposed algorithm(s) [6]. One way to do this is to use an offline dataset like *The Million Song Dataset* [1] and split some part of the data set as training data and run the evaluation on top of the remainder of the data. This approach is meaningful for features that are already available for the dataset, such as e.g. tag prediction for new songs. However, some aspects of recommending songs are inherently subjective, such as serendipity [12], and thus the evaluation of such algorithms can only be done in vivo, i.e. with real users not in an artificial environment.

When conducting an in vivo evaluation, there are some typical issues that need to be considered:

User management. While registering for evaluations, users should be able to provide some context information about them to guide the assignment in groups for A/B testing.

Privacy & Security. User data is highly sensitive, and high standards have to be met wrt. who is allowed to access the data. Also, an evaluation framework needs to ensure that user data cannot be compromised.

Group selection. Users are divided into groups for A/B testing, e.g. based on demographic criteria like age or gender. Then, recommendations for group A are provided by a baseline algorithm, and for group B by the new algorithm.

Playing songs. Unlike other domains, e.g. books, users can give informed decisions by just listening to a song. Thus, to assess a recommended song, it should be possible to play the song directly during the evaluation.

Evaluation monitoring. During an evaluation, it is important to have an overview of how each algorithm performs so far, and how many and how often users participate.

Evaluation metrics. Evaluation results are put into graphs that contain information about the participants and the performance of the evaluated new recommendation algorithm.

Baseline algorithms. Results of an evaluation are often judged by improvements over a baseline algorithm, e.g. a collaborative filtering algorithm [10].

In this paper, we present MUSE – a music recommendation management system – that takes care of all the regular tasks that are involved in conducting an in vivo evaluation. Please note that MUSE can be used to perform in vivo evaluations of *arbitrary* music recommendation algorithms. An instance of MUSE that conforms with state-of-the-art privacy standards is accessible by using the link below, a documentation is available on the MUSE website [2].

muse.informatik.uni-freiburg.de

The remainder of the paper is structured as follows: After a discussion of related work in Section 2, we give an overview of our proposed music recommendation management system in Section 3 with some insights in our evaluation framework in Section 4. Included recommenders are presented in Section 5, and we conclude with an outlook on future work in Section 6.

2. RELATED WORK

The related work is divided in three parts: (1) music based frameworks for recommendations, (2) recommenders' evaluation, (3) libraries and platforms for developing and plugin recommenders.

Music recommendation has attracted a lot of interest from the scientific community since it has many real life applications and bears multiple challenges. An overview

[2] MUSE - Music Sensing in a Social Context: dbis.informatik.uni-freiburg.de/MuSe

of factors affecting music recommender systems and challenges that emerge both for the users' and the recommenders side are highlighted in [17]. Improving music recommendations has attracted equal attention. In [7, 12], we built and evaluated a weighted hybrid recommender prototype that incorporates different techniques for music recommendations. We used Youtube for playing songs but due to a complex process of identifying and matching songs, together with some legal issues, such an approach is no longer feasible. Music platforms are often combined with social media where users can interact with objects maintaining relationships. Authors in [2] leverage this rich information to improve music recommendations by viewing recommendations as a ranking problem.

The next class of related work concerns evaluation of recommenders. An overview of existing systems and methods can be found in [16]. In this study, recommenders are evaluated based on a set of properties relevant for different applications and evaluation metrics are introduced to compare algorithms. Both offline and online evaluation with real users are conducted, discussing how to draw valuable conclusion. A second review on collaborative recommender systems specifically can be found in [10]. It consists the first attempt to compare and evaluate user tasks, types of analysis, datasets, recommendation quality and attributes. Empirical studies along with classification of existing evaluation metrics and introduction of new ones provide insights into the suitability and biases of such metrics in different settings. In the same context, researchers value the importance of user experience in the evaluation of recommender systems. In [14] a model is developed for assessing the perceived recommenders quality of users leading to more effective and satisfying systems. Similar approaches are followed in [3, 4] where authors highlight the need for user-centric systems and high involvement of users in the evaluation process. Relevant to our study is the work in [9] which recognizes the importance for online user evaluation, while implementing such evaluations simultaneously by the same user in different systems.

The last class of related work refers to platforms and libraries for developing and selecting recommenders. The authors of [6] proposed LensKit, an open-source library that offers a set of baseline recommendation algorithms including an evaluation framework. MyMediaLite [8] is a library that offers state of the art algorithms for collaborative filtering in particular. The API offers the possibility for new recommender algorithm's development and methods for importing already trained models. Both provide a good foundation for comparing different research results, but without a focus on in vivo evaluations of music recommenders, thus they don't offer e.g. capabilities to play and rate songs or manage users. A patent in [13] describes a portal extension with recommendation engines via interfaces, where results are retrieved by a common recommendation manager. A more general purpose recommenders framework [5] which is close to our system, allows using and comparing different recommendation methods on provided datasets. An API offers the possibility to develop and incorporate algorithms in the framework, integrate plugins, make configurations and visualize the results. However, our system offers additionally real-time online evaluations of different recommenders, while incorporating end users in the evaluation process. A case study of using Apache Mahout, a library for distributed recommenders based on MapReduce can be found in [15]. Their study provides insights into the development and evaluation of distributed algorithms based on Mahout.

To the best of our knowledge, this is the first system that incorporates such a variety of characteristics and offers a full solution for music recommenders development and evaluation, while highly involving the end users.

3. MUSE OVERVIEW

We propose MUSE: a web-based music recommendation management system, built around the idea of recommenders that can be plugged in. With this in mind, MUSE is based on three main system design pillars:

Extensibility. The whole infrastructure is highly extensible, thus new recommendation techniques but also other functionalities can be added as modular components.

Reusability. Typical tasks required for evaluating music recommendations (e.g. managing user accounts, playing and rating songs) are already provided by MUSE in accordance with current privacy standards.

Comparability. By offering one common evaluation framework we aim to reduce side-effects of different systems that might influence user ratings, improving both comparability and validity of in-vivo experiments.

A schematic overview of the whole system is depicted in Fig. 1. The MUSE Server is the core of our music recommendation management system enabling the communication between all components. It coordinates the interaction with pluggable recommenders, maintains the data in three different repositories and serves the requests from multiple MUSE clients. Next, we will give some insights in the architecture of MUSE by explaining the most relevant components and their functionalities.

3.1 Web-based User Interface

Unlike traditional recommender domains like e-commerce, where the process of consuming and rating items takes up to several weeks, recommending music exhibits a highly dynamic nature raising new challenges and opportunities for recommender systems. Ratings can be given on the fly and incorporated immediately into the recommending process, just by listening to a song. However, this requires a reliable and legal solution for playing a large variety of songs. MUSE benefits from a tight integration of Spotify [3], a music streaming provider that allows listening to millions of songs for free. Thus, recommended songs can be embedded directly into the user interface, allowing to listen and rate them in a user-friendly way as shown in Fig. 2.

[3] A Spotify account is needed to play songs

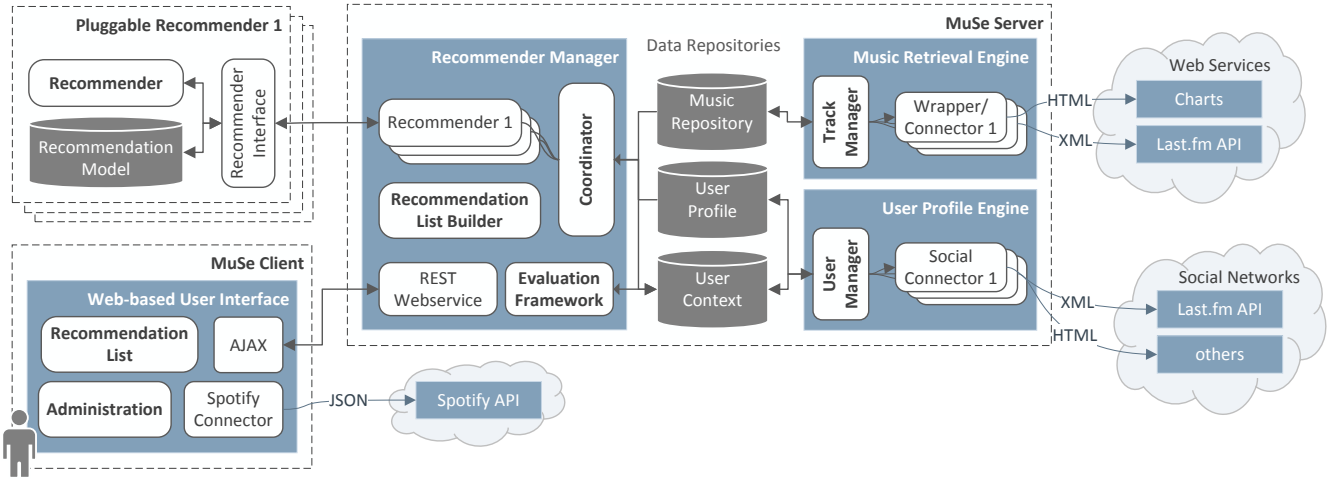

Figure 1. Muse – Music Recommendation Management System Overview

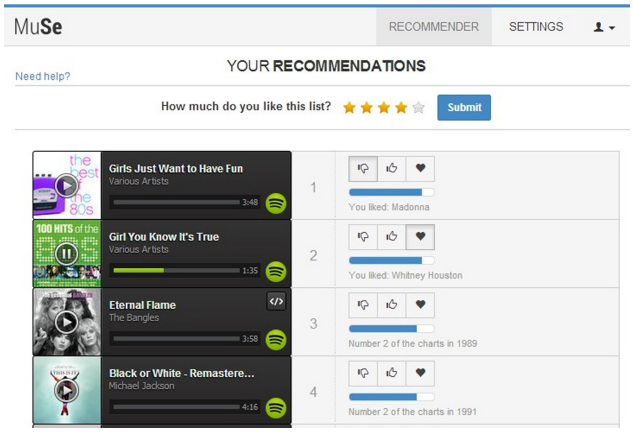

Figure 2. Songs can be played & rated

In order to make sure that users can obtain recommendations without having to be long-time MUSE users, we ask for some contextual information during the registration process. Each user has to provide coarse-grained demographic and preference information, namely the user's spoken languages, year of birth, and optionally a Last.fm user name. In Section 5, we will present five different approaches that utilize those information to overcome the cold start problem. Beyond that, these information is also exploited for dividing users into groups for A/B testing.

Fig. 3 shows the settings pane of a user. Note, that this window is available only for those users, who are not participating in an evaluation. It allows to browse all available recommenders and compare them based on meta data provided with each recommender. Moreover, it is also possible to control how recommendations from different recommenders are amalgamated to one list. To this end, a summary is shown that illustrates the interplay of novelty, accuracy, serendipity and diversity. Changes are applied and reflected in the list of recommendations directly.

3.2 Data Repositories

Although recommenders in MUSE work independently of each other and may even have their own recommendation model with additional data, all music recommenders have access to three global data structures.

The first one is the *Music Repository* that stores songs with their meta data. Only songs in this database can be recommended, played and rated. The *Music Retrieval Engine* periodically collects new songs and meta data from Web Services, e.g. chart lists or Last.fm. It can be easily extended by new sources of information like audio analysis features from the Million Song Dataset [1], that can be requested periodically or dynamically. Each recommender can access all data stored in the Music Repository.

The second repository stores the *User Profile*, hence it also contains personal data. In order to comply with German data privacy requirements only restricted access is granted for both, recommenders and evaluation analyses.

The last repository collects the *User Context*, e.g. which songs a user has listened to with the corresponding rating for the respective recommender.

Access with anonymized user IDs is granted for all recommenders and evaluation analyses. Finally, both user-related repositories can be enriched by the *User Profile Engine* that fetches data from other sources like social networks. Currently, the retrieval of listening profiles of publicly available data from Last.fm and Facebook is supported.

3.3 Recommender Manager

The Recommender Manager has to coordinate the interaction of recommenders with users and the access to the data. This process can be summarized as follows:

- It coordinates access to the repositories, forwards user request for new recommendations, and receives generated recommendations.
- It composes a list of recommendations by amalgamating recommendations from different recommenders into one list based on individual user settings.

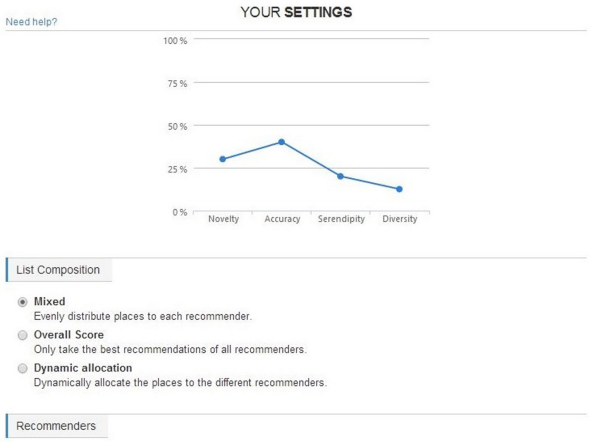

Figure 3. Users can choose from available recommenders

- A panel for administrative users allows enabling, disabling and adding of recommenders that implement the interface described in Section 3.4. Moreover, even composing hybrid recommenders is supported.

3.4 Pluggable Recommender

A cornerstone of MUSE is its support for plugging in recommenders easily. The goal was to design a rather simple and compact interface enabling other developers to implement new recommenders with enough flexibility to incorporate existing approaches as well. This is achieved by a predefined Java interface that has to be implemented for any new recommender. It defines the interplay between the MUSE Recommender Manager and its pluggable recommenders by (1) providing methods to access all three data repositories, (2) forwarding requests for recommendations and (3) receiving recommended items. Hence, new recommenders do not have to be implemented within MUSE in order to be evaluated, it suffices to use the interface to provide a mapping of inputs and outputs [4] .

4. EVALUATION FRAMEWORK

There are two types of experiments to measure the performance of recommenders: (1) offline evaluations based on historical data and (2) in vivo evaluations where users can evaluate recommendations online. Since music is of highly subjective nature with many yet unknown correlations, we believe that in vivo evaluations have the advantage of also capturing subtle effects on the user *during* the evaluation. Since new songs can be rated within seconds by a user, such evaluations are a good fit for the music domain. MUSE addresses the typical issues that are involved in conducting an in-vivo evaluation and thus allows researches to focus on the actual recommendation algorithm.

This section gives a brief overview of how evaluations are created, monitored and analyzed.

[4] More details can be found on our project website.

4.1 Evaluation Setup

The configuration of an evaluation consists of three steps (cf. Fig. 4): (1) A new evaluation has to be scheduled, i.e. a start and end date for the evaluation period has to be specified. (2) The number and setup of groups for A/B testing has to be defined, where up to six different groups are supported. For each group an available recommender can be associated with the possibility of hybrid combinations of recommenders if desired. (3) The group placement strategy based on e.g. age, gender and spoken languages is required. As new participants might join the evaluation over time, an online algorithm maintains a uniform distribution with respect to the specified criteria. After the setup is completed, a preview illustrates how group distributions would resemble based on a sample of registered users.

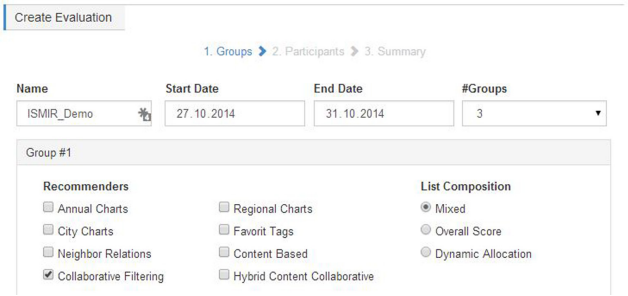

Figure 4. Evaluation setup via Web interface

While an evaluation is running, both registered users and new ones are asked to participate after they login to MUSE. If a user joins an evaluation, he will be assigned to a group based on the placement strategy defined during the setup and all ratings are considered for the evaluation. So far, the following types of ratings can be discerned:

Song rating. The user can provide three ratings for the quality of the recommended song ("love", "like", and "dislike"). Each of these three rating options is mapped to a numerical score internally, which is then used as basis for the analysis of each recommender.

List rating. The user can also provide ratings for the entire list of recommendations that is shown to him on a five-point Likert scale, visualized by stars.

Question. To measure other important aspects of a recommendation like its novelty or serendipity, an additional field with a question can be configured that contains either a yes/no button or a five-point Likert scale.

The user may also decide not to rate some of the recommendations. In order to reduce the number of non-rated recommendations in evaluations, the rating results can only be submitted when at least 50% of the recommendations are rated. Upon submitting the rating results, the user gets a new list with recommended songs.

4.2 Monitoring Evaluations

Running in vivo evaluations as a *black box* is undesirable, since potential issues might be discovered only after the

evaluation is finished. Also, it is favorable to have an overview of the current state, e.g. if there are enough participants, and how the recommenders perform so far. MUSE provides comprehensive insights via an administrative account into running evaluations as it offers an easy accessible visualization of the current state with plots. Thus, adjustments like adding a group or changing the runtime of the evaluation can be made while the evaluation is still running.

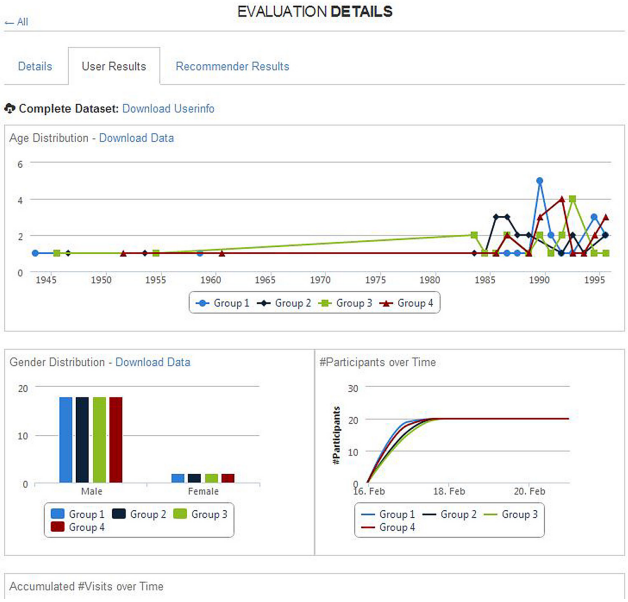

Figure 5. Evaluation results are visualized dynamically

4.3 Analyzing Evaluations

For all evaluations, including running and finished ones, a result overview can be accessed that shows results in a graphical way to make them easier and quicker to grasp (c.f. Fig. 5). The plots are implemented in a dynamic fashion allowing to adjust, e.g., the zoom-level or the displayed information as desired. They include a wide range of metrics like group distribution, number of participants over time, averaged ratings, mean absolute error, accuracy per recommender, etc. Additionally, the complete dataset or particular plotting data can be downloaded in CSV format.

5. RECOMMENDATION TECHNIQUES

MUSE comes with two types of recommenders out-of-the-box. The first type includes traditional algorithms, i.e. *Contend Based* and *Collaborative Filtering* [10] that can be used as baseline for comparison. The next type of recommenders is geared towards overcoming the cold start problem by (a) exploiting information provided during registration (*Annual*, *Country*, and *City Charts* recommender), or (b) leveraging knowledge from social networks (*Social Neighborhood* and *Social Tags* recommender).

Annual Charts Recommender. Studies have shown, that the apex of evolving music taste is reached between the age of 14 and 20 [11]. The Annual Charts Recommender exploits this insight and recommends those songs, which were popular during this time. This means, when a user indicates 1975 as his year of birth, he will be assigned to the music context of years 1989 to 1995, and obtain recommendations from that context. The recommendation ranking is defined by the charts position in the corresponding annual charts, where the following function is used to map the charts position to a score, with c_s as the position of song s in charts c and n is the maximum rank of charts c:

$$score(s) = -log(\frac{1}{n}c_s) \quad (1)$$

Country Charts Recommender. Although music taste is subject to diversification across countries, songs that a user has started to listen to and appreciate oftentimes have peaked in others countries months before. This latency aspect as well as an inter-country view on songs provide a good foundation for serendipity and diversity. The source of information for this recommender is the spoken languages, provided during registration, which are mapped to a set of countries for which we collect the current charts. Suppose there is a user a with only one country A assigned to his spoken languages, and C_A the set of charts songs for A. Then, the set C_R of possible recommendations for a is defined as follows, where L is the set of all countries:

$$C_R = (\bigcup_{X \in L} C_X) \setminus C_A$$

The score for a song $s \in C_R$ is defined by the average charts position across all countries, where Function (1) is used for mapping the charts position into a score.

City Charts Recommender. While music tastes differ across countries, they may likewise differ across cities in the same country. We exploit this idea by the City Charts Recommender, hence it can be seen as a more granular variant of the Country Charts Recommender. The set of recommendations C_R is now composed based on the city charts from those countries a user was assigned to. Hereby, the ranking of songs in that set is not only defined by the average charts position, but also by the number of cities where the song occurs in the charts: The fewer cities a song appears in, the more "exceptional" and thus relevant it is.

Social Neighborhood Recommender. Social Networks are, due to their growing rates, an excellent source for contextual knowledge about users, which in turn can be utilized for better recommendations. In this approach, we use the underlying social graph of Last.fm to generate recommendations based on user's Last.fm neighborhood which can be retrieved by our *User Profile Engine*. To compute recommendations for a user a, we select his five closest neighbors, an information that is estimated by Last.fm internally. Next, for each of them, we retrieve its recent top 20 songs and thus get five sets of songs, namely $N_1 ... N_5$. Since that alone would provide already known songs in general, we define the set N_R of possible recommendations as follows, where N_a is the set of at most 25 songs a

user a recently listened to and appreciated:

$$N_R = (\bigcup_{1 \leq i \leq 5} N_i) \setminus N_a$$

Social Tags Recommender. Social Networks collect an enormous variety of data describing not only users but also items. One common way of characterising songs is based on tags that are assigned to them in a collaborative manner. Our Social Tag Recommender utilizes such tags to discover new genres which are related to songs a user liked in the past. At first, we determine his recent top ten songs including their tags from Last.fm. We merge all those tags and filter out the most popular ones like "rock" or "pop" to avoid getting only obvious recommendations. By counting the frequency of the remaining tags, we determine the three most common thus relevant ones. For the three selected tags, we use again Last.fm to retrieve songs where the selected tags were assigned to most frequently.

To test our evaluation framework as well as to assess the performance of our five recommenders we conducted an in vivo evaluation with MUSE. As a result 48 registered users rated a total of 1567 song recommendations confirming the applicability of our system for in vivo evaluations. Due to space limitations, we decided to omit a more detailed discussion of the results.

6. CONCLUSION

MUSE puts the fun back in developing new algorithms for music recommendations by taking the burden from the researcher to spent cumbersome time on programming yet another evaluation tool. The module-based architecture offers the flexibility to immediately test novel approaches, whereas the web-based user-interface gives control and insight into running in vivo evaluations. We tested MUSE with a case study confirming the applicability and stability of our proposed music recommendation management system. As future work, we envision to increase the flexibility of setting up evaluations, add more metrics to the result overview, and to develop further connectors for social networks and other web services to enrich the user's context while preserving data privacy.

7. REFERENCES

[1] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR*, 2011.

[2] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang 0005, and Xiaofei He. Music recommendation by unified hypergraph: combining social media information and music content. In *ACM Multimedia*, pages 391–400, 2010.

[3] Li Chen and Pearl Pu. User evaluation framework of recommender systems. In *Workshop on Social Recommender Systems (SRS'10) at IUI*, volume 10, 2010.

[4] Paolo Cremonesi, Franca Garzotto, Sara Negro, Alessandro Vittorio Papadopoulos, and Roberto Turrin. Looking for "good" recommendations: A comparative evaluation of recommender systems. In *INTERACT (3)*, pages 152–168, 2011.

[5] Aviram Dayan, Guy Katz, Naseem Biasdi, Lior Rokach, Bracha Shapira, Aykan Aydin, Roland Schwaiger, and Radmila Fishel. Recommenders benchmark framework. In *RecSys*, pages 353–354, 2011.

[6] Michael D. Ekstrand, Michael Ludwig, Joseph A. Konstan, and John Riedl. Rethinking the recommender research ecosystem: reproducibility, openness, and LensKit. In *RecSys*, pages 133–140, 2011.

[7] Simon Franz, Thomas Hornung, Cai-Nicolas Ziegler, Martin Przyjaciel-Zablocki, Alexander Schätzle, and Georg Lausen. On weighted hybrid track recommendations. In *ICWE*, pages 486–489, 2013.

[8] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Mymedialite: a free recommender system library. In *RecSys*, pages 305–308, 2011.

[9] Conor Hayes and Pádraig Cunningham. An on-line evaluation framework for recommender systems. *Trinity College Dublin, Dep. of Computer Science*, 2002.

[10] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. In *ACM Trans. Inf. Syst.*, pages 5–53, 2004.

[11] Morris B Holbrook and Robert M Schindler. Some exploratory findings on the development of musical tastes. *Journal of Consumer Research*, pages 119–124, 1989.

[12] Thomas Hornung, Cai-Nicolas Ziegler, Simon Franz, Martin Przyjaciel-Zablocki, Alexander Schätzle, and Georg Lausen. Evaluating Hybrid Music Recommender Systems. In *WI*, pages 57–64, 2013.

[13] Stefan Liesche, Andreas Nauerz, and Martin Welsch. Extendable recommender framework for web-based systems, 2008. US Patent App. 12/209,808.

[14] Pearl Pu, Li Chen, and Rong Hu. A user-centric evaluation framework for recommender systems. In *RecSys '11*, pages 157–164, New York, NY, USA, 2011. ACM.

[15] Carlos E Seminario and David C Wilson. Case study evaluation of mahout as a recommender platform. In *RecSys*, 2012.

[16] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297, 2011.

[17] Alexandra L. Uitdenbogerd and Ron G. van Schyndel. A review of factors affecting music recommender success. In *ISMIR*, 2002.