EVALUATING MUSIC RECOMMENDATION IN A REAL-WORLD SETTING: ON DATA SPLITTING AND EVALUATION METRICS

Szu-Yu Chou, Yi-Hsuan Yang, and Yu-Ching Lin

Research Center for IT innovation, Academia Sinica, Taipei, Taiwan
{fearofchou,yang}@citi.sinica.edu.tw, aaronlin@kkbox.com

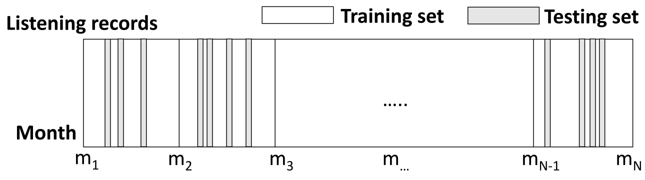
ABSTRACT

Evaluation is important to assess the performance of a computer system in fulfilling a certain user need. In the context of recommendation, researchers usually evaluate the performance of a recommender system by holding out a random subset of observed ratings and calculating the accuracy of the system in reproducing such ratings. This evaluation strategy, however, does not consider the fact that in a real-world setting we are actually given the observed ratings of the past and have to predict for the future. There might be new songs, which create the cold-start problem, and the users' musical preference might change over time. Moreover, the user satisfaction of a recommender system may be related to factors other than accuracy. In light of these observations, we propose in this paper a novel evaluation framework that uses various time-based data splitting methods and evaluation metrics to assess the performance of recommender systems. Using millions of listening records collected from a commercial music streaming service, we compare the performance of collaborative filtering (CF) and content-based (CB) models with low-level audio features and semantic audio descriptors. Our evaluation shows that the CB model with semantic descriptors obtains a better trade-off among accuracy, novelty, diversity, freshness and popularity, and can nicely deal with the cold-start problems of new songs.

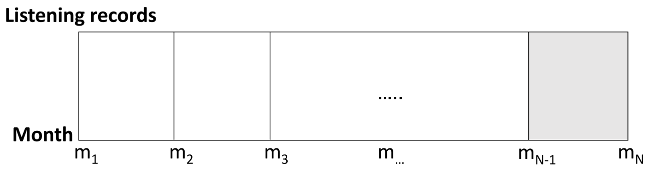
Index Terms— Collaborative filtering, content-based recommendation, cold-start, data splitting, evaluation metrics

1. INTRODUCTION

Online music streaming service providers such as KKBOX[1], Spotify[2] and Grooveshark[3] offer a large number of tracks of different styles to meet the tastes of users from different cultures and backgrounds. To help users find the music they want, music recommender systems have been developed, based on for example collaborative filter (CF) and content-based (CB) models [1–11]. Such systems use either explicit user ratings (e.g. star ratings) or implicit ratings (e.g. playcounts) to learn the preference of users. The evaluation is typically done by randomly splitting observed ratings into training and test sets, learning the model from the training set and then calculating the accuracy in predicting the ratings of the test set. While early work focuses on only accuracy, increasing attention has been made to evaluate a recommender system in terms of other measures such as diversity and novelty [12–14]. User-centered evaluation methods have also been advocated [15].



(a) Traditional split strategy (TS)



(b) Real-life split strategy (RLS)

Fig. 1: Strategies for splitting data into training and test sets.

It is important, though, to note that it is usually easier to record the listening records instead of explicit ratings from the users of a music streaming service [5]. Given the listening records of a number of users across a span of time, the aforementioned data splitting strategy, which we refer to as the *traditional split strategy* (TS), would create a user-item rating matrix that records how often a user listens to a track by summing up the playcounts over that span of time. The time information of the listening records is thereby neglected. This has a number of important limitations, because without the time information we cannot investigate issues such as how the user's preference changes over time [16], how the popularity of a track changes, and how we should respond to newly released songs. Moreover, in a real-world setting, a recommender system usually needs to make predictions for the near future instead for the past.

As the most direct treatment of this issue, Benett and Lam-

This work is funded by the Ministry of Science and Technology, Taiwan.

[1]`https://www.kkbox.com`
[2]`https://www.spotify.com`
[3]`http://grooveshark.com/`

ing [4] proposed to split the data by time, using the most recent listening records as the test set and the remaining data set for training (see Fig. 1). We refer to this strategy as the *real-life split strategy* (RLS), for it better reflects practical situations. This is an important step forward. However, we note that in RLS, the test set may be composed of a mixture of old and new songs. For those new songs that have never been listened to in the training set, there is a so-called *cold-start* problem [11] for which the widely-used CF models cannot work well. Benett and Laming did not pay attention to this issue and only used CF-based models in their system [4].

Noting that the data splitting strategy influences how we assess the performance of a recommender system, we propose a novel evaluation framework that considers up to four data splitting strategies, including TS and three variants of RLS, that place different emphases on the cold start problem (Section 3.2). Moreover, using millions of listening records collected from a commercial music streaming service (Section 3.1), we systematically compare the performance of CF and CB models (Section 4.1) using two low-level audio features and three types of high-level semantic audio descriptors (Sections 4.2 and 4.3). In addition to accuracy, we propose measurements of *novelty*, *diversity*, *freshness* and *popularity* (Section 3.3), so as to gain more insights into the performance of a recommender system. To the best of our knowledge, this represents one of the most comprehensive evaluations of a music recommender system in a real-world setting.

Our evaluation shows that the performance of a recommender system varies according to the adopted data splitting strategy and evaluation criterion. While the CF models lead to high accuracy for TS, it cannot work for a number of RLS settings, and it tends to recommend popular songs. CB models using low-level audio features can deal with the cold-start problem, and its recommendation is rich in diversity, but better tradeoff between accuracy, diversity and the other performance measurements is obtained by making use of the semantic descriptors and by hybridizing CF and CB models.

2. RELATED WORKS

A large number of recommendation models have been proposed in the literature [1–11]. The advantage of CF models is that they can adequately leverage the collective wisdom of user preference from the listening records of a number of users and then make recommendations. However, it has also been known that CF models cannot deal with new songs (because there is not corresponding information in the training set) and that the recommended items of CF models may lack novelty [17]. CB models, on the other hand, makes use of the content features of the items and recommend songs that are similar in content with those songs favored by the user in the training set. Therefore, CB models can deal with new songs, but they cannot learn from the preference of other users. Due to their complementary nature, hybridizing CF and CB models can usually lead to better recommendation performance [11].

In addition to the RLS strategy, there are other ways to take time information into account. For example, Ding and Li [3] decreased the weight of old data while creating the user-item rating matrix; Koren [8] considered the timestamps of listening records as an additional feature while learning the model. To learn user preference from listening records of streaming services, it is also possible to use the dwell time of listening (e.g. whether the user finished listening to that song or skipped it in the middle) as the learning target [10], or to take into account both observed and unobserved ratings [5]. While these directions are also promising, we opt for focusing on data splitting strategies and evaluation metrics in this work.

3. A REAL-WORLD SETTING FOR EVALUATION

3.1. The KKBOX dataset

KKBOX is a leading music streaming service provider in East Asia. From KKBOX we obtain a real-world dataset comprising around 0.1 billion listening records from October 2012 to September 2013 (totally 12 months). It covers around 28k users and 124k songs. Each of the listening records contains the listening timestamp, song title, artist name, album name and genre labels selected from a predefined set. Depending on the data splitting strategy, we would count the number of listening records for each song for each user for the training and test sets respectively. Following [5], we transform the resulting playcounts into *confidence values* according to the following formula: $c_{ui} = 1 + \alpha \log(1 + y_{ui}/\varepsilon)$, where y_{ui} is the playcount of the u-th user for the i-th song, c_{ui} is the confidence value, and α and ε are user-defined hyperparameters, which are both set to 1 in this work. In other words, we would use c_{ui} instead of y_{ui} as the data in both the training and evaluation of the recomender model. This is preferable because the original playcounts may contain outliers and noises [5].

While there are many other widely-used music recommendation datasets in the public domain, such as the Celma 360K, MSD or the Yahoo! datasets [18], we opt for using this KKBOX dataset because we have access to the audio files for extracting various audio features due to a partnership with the company, and because KKBOX is an active commercial music streaming service provider.

3.2. Data splitting

We consider the following data splitting strategies.

- **Traditional split strategy (TS):** This strategy randomly splits data into training and test sets (cf. Fig. 1(a)). Because of the randomization, it is possible that all the songs would appear in both the training and test sets. Therefore, the corresponding evaluation problem can be considered as a warm-start one (all the songs in test set have been seen before). This is the most commonly used data splitting strategy in the literature.

Table 1: The statistics of the training and test sets

Data sets	#users	#songs	#entries
TS Train	27k	122k	4,603k
TS Test	25k	49k	672k
RLSc Train	27k	121k	4,604k
RLSc Test	4k	3k	15k
RLSw Train	27k	121k	4,604k
RLSw Test	17k	48k	691k
RLSa Train	27k	121k	4,604k
RLSa Test	17k	51k	707k

- **Real-life split strategy (RLS):** This strategy uses the data in the most recent month as the test set, and the data in the remaining months (i.e. the past 11 moths in our case) as the training set. Therefore, it is a time-based splitting (cf. Fig. 1(b)). Because there might be new songs or new users in the most recent month, there is a cold-start problem. We do not consider the new users and focus on only the new songs in this study. Specifically, while the training set is fixed, we consider three variants of the test set in RLS. (1) The first one, RLSc, considers only those new songs in the test set, leading to an extremely *cold*-start recommendation problem. (2) The second one, RLSw, requires that the songs in the test set has been observed in the training set, leading to a purely *warm*-start problem. (3) The test songs of the final RLSa setting is the union of *all* the test songs of the above two. Please see Table 1 for the data statistics of the four splitting strategies.

3.3. Evaluation metrics

We consider two measures to evaluate the accuracy of recommendation: the root mean square error (RMSE) and normalized discounted cumulative gain (NDCG).

- **Accuracy:** RMSE measures closeness between predicted rating and the true rating. Given n entries of test data across all the users, RMSE is calculated by using the squared difference between the true confidence value c_j for the j-th entry and the corresponding estimate $\hat{c}_j$ [11]. Smaller RMSE indicates better accuracy. NDCG differs from RMSE in two aspects. First, it first evaluates the accuracy of the recommender system for each user and then gets the average. Second, it formulates recommendation as a ranking problem, generates a ranked list of songs in descending order of the relevance of the songs to a particular user, and then evaluates the similarity between the predicted ranking with the true ranking observed from data [10]. In this work, we consider only the top k songs in the ranking list. Larger NDCG indicates better accuracy. It can be found that either RMSE or NDCG is computed using only the songs that have actually been listened to by the user. Depending on how we split the training and test sets, we can evaluate the RMSE or NDCG accordingly.

We also evaluate the performance of the recommender system in terms of novelty, diversity, freshness and popularity through recommending top-N songs for each user. Because of the evaluation of these measures does not require observed ratings, the top-N songs are not restricted to those songs that have actually been listened to by the user.

- **Novelty:** We want to evaluate whether the top recommended songs are new to the user or not. It would be interesting if the user is recommended with songs that are outside of the user's musical comfort zone. We evaluate this in the artist level. Given UA_u, the set of artists that the user has listened to, and RA_u, the set of artists in the top-N songs recommended to the user, we define novelty as follows

$$novelty_u = \frac{|RA_u \setminus UA_u|}{|RA_u|} \,. \qquad (1)$$

We compute this value for this user and take the average $\frac{1}{U}\sum_{u=1}^{U} novelty_u$ as the measurement of novelty, where U denotes the number of users.

- **Diversity:** This measurement evaluates how diverse the top-N songs are in terms of the genre of the songs. Specifically, we look up the genre information provided by the KKBOX company and compute the genre histogram h_u. The diversity is defined as the average entropy of the genre histogram across the users:

$$diversity = -\frac{1}{U}\sum_{u=1}^{U}\sum_{g=1}^{G} h_u(g)\log(h_u(g)) \,, \qquad (2)$$

where $h_u(g)$ denotes the number of songs for the g-th genre and G is the total number of genres.

- **Freshness:** We measure the freshness by taking the average of the release date of the recommended songs by referring to the information provided by the KKBOX company. It reflects whether the top-N songs recommended to the user are composed of new or old songs. Again, we compute the freshness across all the users.

$$freshness = \frac{1}{U}\frac{1}{N}\sum_{u=1}^{U}\sum_{i=1}^{N} RD_{ui} \,, \qquad (3)$$

where RD_{ui} denotes the timestamp of the i-th song for the u-th user in UNIX time (number of seconds that have elapsed since 00:00:00 UTC, 1 January 1970).

- **Popularity:** We measure the popularity of a song according to PS_{ui}, the number of users that have listened to that song. Accordingly, we can evaluate the popularity of recommendation by taking the average across the top-N songs for the U users.

$$popularity = \frac{1}{U}\frac{1}{N}\sum_{u=1}^{U}\sum_{i=1}^{N} PS_{ui} \,. \qquad (4)$$

Table 2: The low-level audio features and high-level semantic audio descriptors adopted in this work

Feature type	Feature	Dim.	Description
Low-level	MFCC	120	Pooled Mel-frequency cepstral coefficients and their 1st and 2nd differences
	AW	8,192	Audio words computed by applying sparse coding on log-scale magnitude spectra
High-level	AWA	435	AW-based prediction of 435 acoustic tags defined by CAL10k [19]
	AWG	140	AW-based prediction of 140 genre tags defined by CAL10k [19]
	AWM	190	AW-based prediction of 190 emotion tags defined by MER31k [20]

We note that these performance measures evaluate the recommender system in different aspects, and that the combination of them may be considered as an estimate of the user satisfaction of the system. While similar concepts have been proposed in the literature [14], relatively little work has been done to consider all these measures in a real-world setting.

4. RECOMMENDER SYSTEM

4.1. Recommendation algorithm

Due to the superior performance of factorization-based models [6, 7], we consider the following three models in this work. The former two are CF models, whereas the last one can be either CF, CB, or hybrid depending on the features we use.

- **Matrix Factorization (MF)** [6] learns the user latent factor $\boldsymbol{p}_u$ and item latent factor $\boldsymbol{q}_i$ from the user-item rating matrix by solving the following optimization problem: $\min_{\boldsymbol{q}*,\boldsymbol{p}*} \sum_{u,i}(c_{ui}-\boldsymbol{q}_i^T\boldsymbol{p}_u)^2+\lambda(\|\boldsymbol{q}_i\|^2+\|\boldsymbol{p}_u\|^2)$, where λ is a regularization parameter to avoid overfitting and c_{ui} is the target value of the prediction.
- **SVD++** [7] extends MF by integrating implicit feedback and bias of rating into the latent factor model. Specifically, its prediction formula is: $\hat{c}_{ui} = \mu + b_u + b_i + \boldsymbol{q}_i^T \left(\boldsymbol{p}_u + |R_u|^{-1/2}\sum_{j\in R_u} c_i\right)$, where μ is the global mean rating, b_u and b_i are the observed biases, and R_u is the set of implicit information.
- **Factorization machine (FM)** [21] is a feature-based MF model. Its main idea is to learn the interactions between all the feature variables. It is generally defined as: $\hat{c} = w_0 + \sum_{i=1}^m w_i x_i + \sum_{i=1}^m \sum_{j=i+1}^m \hat{w}_{ij} x_i x_j$, where w_0 is the global bias parameter, m is the number of features, w_i is the weight of features x_i, and $\hat{w}_{ij}$ is a measure of the second-order interaction between feature variables x_i and x_j. In our implementation, we use the the MCMC solver of the libFM toolkit for FM [21].

4.2. Low-level audio features

Many audio features have been proposed in the literature. The features can be either computed by *feature design* using musical knowledge and signal processing, or by *feature learning*, which learns a set of features by machine learning. In our work, we consider the Mel-frequency cepstral coefficients (MFCC) as a representative feature from feature design, and the audio word (AW) based features for feature learning.

- **MFCC** is arguably the most widely-used feature in audio signal processing. We compute 20 MFCCs from windows of 1,024 audio frames, corresponding to 30 ms at a sampling rate of 22 kHz, and a hop size of 512 samples. We also compute the first and second order differences and take the mean and standard deviation for pooling, leading to a 120-D feature vector
- **AW:** The classic vector quantization approach represents a series of low-level audio feature vectors as the counts of the occurrence of codewords in a pre-built dictionary, leading to a histogram that has been referred to as an AW representation [9]. An AW representation is characterized by its ability of symbolizing local audio events as a codeword, its flexibility of using an arbitrary large number of codewords learned from a corpus of audio data in an unsupervised fashion, and its light dependence on domain knowledge for feature design. Therefore, it has been considered as a powerful alternative to conventional audio features. In our implementation, we use the online dictionary learning algorithm [22] to optimize the dictionary, sparse coding for codeword assignment [23], and log-scale magnitude spectra as the underlying raw audio features. Polar split rectification and cubic-root power normalization is applied for post processing. This pipeline for AW computation is found effective in our prior work [24].

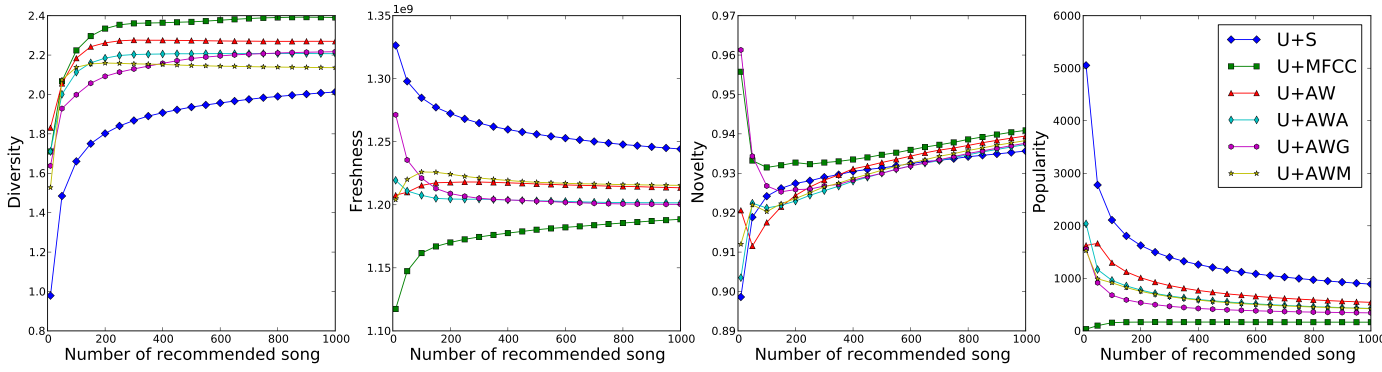
4.3. High-level Semantic audio descriptors

While low-level audio features might not be able to well represent our perception of music, we employ a number of music auto-taggers to compute the three types semantic audio descriptors from audio signals: *genre*, *acoustics*, and *emotion*. The auto-taggers are trained by using the following two datasets.

- **CAL10k** is a music dataset that contains expert annotations of 10k songs for 435 acoustic-related and 140 genre tags [19]. The acoustic tags include those related to the instruments, timbre of singing voice, and acoustic qualities of the music. The genre tags include 140 types of genres and sub-genres that are finer in granularity comparing to those provided by KKBOX.
- **MER31k** is another music dataset that contains the emotion labeling over 190 possible emotion tags over 31,427

Table 3: Accuracy (in RMSE and NDCG@10) of different models and features for different data splitting settings. Please note that 'U' and 'S' denote the user IDs and song IDs, respectively, and 'U+S' corresponds to the conventional CF data

Model	Feature	RMSE				NDCG@10			
		TS	RLSc	RLSw	RLSa	TS	RLSc	RLSw	RLSa
CF-MF	U+S	1.2600	–	1.2600	–	0.5950	–	0.6048	–
CF-SVD++	U+S	0.8192	–	0.7766	–	0.6841	–	0.6704	–
CF-FM	U+S	**0.7547**	–	0.7369	–	**0.7344**	–	0.7046	–
CB-FM	U+MFCC	0.8001	0.6907	0.7346	0.7337	0.6576	0.6471	0.6341	0.6335
	U+AW	0.7830	0.6869	0.7327	0.7317	0.7081	0.6543	0.6851	0.6843
	U+AWA	0.7885	**0.6852**	**0.7309**	**0.7300**	0.6985	**0.6689**	0.6878	0.6869
	U+AWG	0.7914	0.6889	0.7316	0.7307	0.6902	0.6436	0.6762	0.6754
	U+AWM	0.7901	0.6898	0.7319	0.7310	0.6918	0.6446	0.6788	0.6776
Hybrid-FM	U+S+MFCC	0.7570	0.6907	0.7362	0.7349	0.7323	0.6471	0.7028	0.7021
	U+S+AW	0.7553	0.6869	0.7367	0.7353	0.7336	0.6543	0.7032	0.7026
	U+S+AWA	0.7550	**0.6852**	0.7371	0.7357	0.7328	**0.6689**	0.7043	0.7036
	U+S+AWG	**0.7547**	0.6889	0.7359	0.7346	0.7331	0.6436	**0.7056**	**0.7049**
	U+S+AWM	0.7552	0.6898	0.7368	0.7354	0.7322	0.6446	0.7037	0.7031



(a) Diversity (b) Freshness (c) Novelty (d) Popularity

Fig. 2: The performance of different FM-based models for top-N recommendation, measured in four evaluation criteria.

songs, annotated by the crowd of the users of a social music platform as a result of social tagging [20]. Only the songs that are highly labeled with any of these emotion tags are considered.

We train auto-taggers for labeling all the songs in the KKBOX dataset by using MER31k for emotion tags and CAL10k for acoustic and genre tags. The auto-taggers are trained by using AW as the feature representation and linear support vector machines as classifiers. We have reported in [24] that such a setting leads to fairly accurate estimation.

5. EXPERIMENTAL

5.1. Evaluation on accuracy

Table 3 shows the accuracy of different models and features for different data splitting settings. From the first three rows, we see that CF-FM outperforms CF-MF and CF-SVD++ in TS and RLSw, the two warm-start settings, in both RMSE and NDCG, showing the effectiveness of FM. Next, from the result of CB-FM, we see that although CB models are inferior to CF models for TS and RLSw, they can perform quite well for the two cold-start settings RLSc and RLSa. The high-level features seem to perform slightly better than the low-level features. Moreover, better results for all the four data splitting settings can be obtained by using a hybrid model that linearly combines the estimates of CF and CB models, as the last few rows show. If we only consider TS, we would not know that CF might not work consistently well in a real-life setting.

5.2. Evaluation on other performance measures

In this evaluation, we train a recommender system using the RLS training set and evaluate the top-N recommended songs for 1,000 randomly selected users. Results shown in Fig. 2 confirms that the recommendation of CF-FM (i.e. U+S) lacks novelty [17]. Moreover, we see that the recommendation of CF-FM has highest popularity, highest freshness and lowest diversity, possibly because many users tend to listen to popular or newly released songs. On the other hand, we see that CB-FM model with MFCC has the highest diversity and novelty, and the lowest freshness and popularity. It seems that this model attempts to recommend long-tail and older songs. In contrast, the CB-FM model with semantic audio descriptors generally obtains a better balance between the four performance measures, which might eventually lead to higher user satisfaction. Although further user studies are needed, this evaluation provides more insights into the performance of these models.

6. CONCLUSIONS

In this paper, we have proposed a novel framework for evaluating the performance of a music recommender system in a real-world setting. We have also presented a comprehensive evaluation of various CF and CB models, low- and high-level audio features using the proposed framework, with a real-world dataset and some novel performance measures. We show that factorization machine based CB models perform well for a number of real-life data splitting settings that involve cold-start problems, and that using high-level semantic features (e.g. genre and emotion) in CB achieves a good trade-off between the accuracy, diversity, novelty and other measures of recommendation. This work confirms the importance of proper evaluation protocols of a recommender system. The evaluation framework is generic and hopefully can be applied to other recommendation problems as well. Future work can be directed to the reproducibility of evaluations and the involvement of real users in the evaluation cycle.

7. REFERENCES

[1] B. Sarwar et al., "Item-based collaborative filtering recommendation algorithms," in *Proc. WWW*, 2001, pp. 285–295.

[2] Q. Li et al., "A music recommender based on audio features," in *Proc. ACM SIGIR*, 2004, pp. 532–533.

[3] Y. Ding and X. Li, "Time weight collaborative filtering," in *Proc. ACM CIKM*, 2005, pp. 485–492.

[4] J. Bennett and S. Lanning, "The netflix prize," in *Proc. KDD Cup Workshop*, 2007.

[5] Y. Hu et al., "Collaborative filtering for implicit feedback datasets," in *Proc. ICDM*, 2008, pp. 263–272.

[6] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. ACM SIGKDD*, 2008, pp. 426–434.

[7] Y. Koren et al., "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[8] Y. Koren, "Collaborative filtering with temporal dynamics," *CACM*, vol. 53, no. 4, pp. 89–97, 2010.

[9] B. Mcfee et al., "Learning content similarity for music recommendation," *IEEE TASLP*, 2012.

[10] X. Yi et al., "Beyond clicks: Dwell time for personalization," in *Proc. ACM Recsys*, 2014, pp. 113–120.

[11] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proc. ACM MM*, 2014, pp. 627–636.

[12] S. M. McNee et al., "Being accurate is not enough: How accuracy metrics have hurt recommender systems," in *Pros. ACM CHI*, 2006, pp. 1097–1101.

[13] S. Vargas et al., "Rank and relevance in novelty and diversity metrics for recommender systems," in *Proc. ACM Recsys*, 2011, pp. 109–116.

[14] L. Shi, "Trading-off among accuracy, similarity, diversity, and long-tail: A graph-based recommendation approach," in *Proc. ACM Recsys*, 2013, pp. 57–64.

[15] S. Dooms, T. De Pessemier, and L. Martens, "A user-centric evaluation of recommender algorithms for an event recommendation system," in *Proc. Works. IntRS*, 2011.

[16] L. Xiang et al., "Temporal recommendation on graphs via long- and short-term preference fusion," in *Proc. ACM SIGKDD*, 2010, pp. 723–732.

[17] A. Oord et al., "Deep content-based music recommendation," in *Proc. NIPS*, pp. 2643–2651. 2013.

[18] E. Zangerle et al., "#nowplaying music dataset: Extracting listening behavior from twitter," in *Proc. Int. Works. ISMM*, 2014.

[19] D. Tingle et al., "Exploring automatic music annotation with "acoustically-objective" tags," in *Proc. MIR*, 2010.

[20] Y.-H. Yang and J.-Y. Liu, "Quantitative study of music listening behavior in a social and affective context," *TMM*, vol. 15, no. 6, pp. 1304–1315, Oct 2013.

[21] R. Steffen, "Factorization machines with libfm," *ACM TIST*, vol. 3, no. 3, pp. 57:1–57:22, May 2012.

[22] J. Mairal et al., "Online dictionary learning for sparse coding," in *ICML*, 2009, pp. 689–696.

[23] F. Bach et al., "Optimization with sparsity-inducing penalties," *FTML*, 2012.

[24] C.-C. M. Yeh et al., "Awtoolbox: Characterizing audio information using audio words," in *Proc. ACM MM*, 2014, pp. 809–812.