MUSIC RECOMMENDATION BASED ON ARTIST NOVELTY AND SIMILARITY

Paper ID: 550

ABSTRACT

There is an increasing interest in the music community to promote and discover new talents. Unlike most previous systems that recommend songs that are already popular or are sung by popular artists, the system proposed in this paper recommends songs that are new to the user. In addition, these songs are sung by independent singers or artists whose popularity is on the rise. An artist new to the user is selected based on the similarity between the artist and the users favorite artists. The performance of the proposed system is evaluated using the Spotify Radio Recommender as a reference and a pool of 100 subjects recruited on campus. Experimental results show that our system achieves a high novelty score and a competitive user-preference score.

Index Terms— music recommendation system, novelty, similarity

1. INTRODUCTION

Online music is abundant nowadays. Thanks to the advance of internet technology, it is relatively easy to access music. However, finding ones favorite music from millions of songs is often not as easy as one would like. From user's perspective, it is desirable that online music providers automatically recommend songs that match users expectation or taste. Music recommendation guided by artist similarity appears to be a plausible approach. Indeed, many services of this kind such as those provided by Spotify [1], Last.fm, iTunes [2], etc. have become more and more popular [**?**]. Given the favorite artists input by the user, the Spotify Radio identifies additional artists similar to the favorite artists and recommends a playlist that contains songs of both kinds of artists. Last.fm identifies similar artists based on the users' listening history. Similarly, iTunes Ping finds similar artists from the users' libraries and preference. However, existing recommendation systems tend to select most popular rather than less or not popular artists. This leads to low-novelty recommendations. Moreover, artists with low popularity may become undiscovered forever. In this paper, we propose a new system aiming at recommending artists that are new to users and promoting new talents. Our approach is extremely effective in recommending novel music to surprise users and satisfy their needs. It works for all music preferences ranging from pop, electronic, metal, jazz, rock, hip hop, country, hardcore, to vocal music. The system is robust and efficient. It is capable of discovering new artists for users based on a small input artist list in a short period of time. The rest of this paper is organized as follows. In Section 2, we briefly review two major approaches to music recommendation. Section 3 describes our recommendation system in details. The performance evaluation of the recommendation system is reported in Section 4. Finally, Section 5 concludes the paper and discusses future work.

2. RELATED WORK

Over the past few years, two types of recommendation techniques have been developed: Content-based and collaborative filtering (CF). This section reviews these techniques.

2.1. Content-based Filtering

The basic assumption of the content-based filtering approach [3] is that items (i.e., song or artist) with similar properties are equally attractive to a user. It compares various items unknown to the user with those previously rated by the user, and the ones with the best similarity scores are recommended [4]. Acoustic properties of songs such as rhythm, tempo, frequency spectrum, genre, etc. are often used for similarity comparison. A major advantage of content-based filtering is that it is free of the new item problem, which refers to the inability of a music recommendation system to recommend new items that have not been rated by any user. Therefore, new artists and artists who are already popular both have equal chance to be recommended. However, user independence is a major drawback of content-based filtering, because it focuses on a specific user without considering the listening histories of other users. As a result, systems based on this approach suffer from over-specialization and are unable to recommend items other than those already in the users listening history.

2.2. Collaborative Filtering

Collaborative filtering (CF) predicts a user's interest from a large collection of preferences or taste profiles of other users. The basic idea of CF is that if two users have similar tastes, they may enjoy each other's favorite songs [5]. Based on the

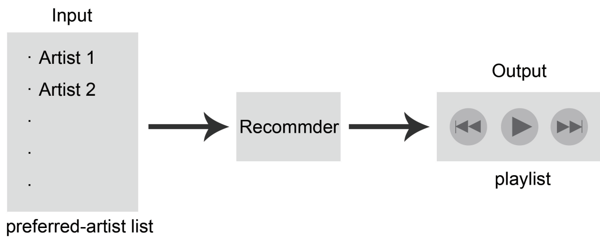

Fig. 1. System Overview

similarity in users' rating or historical behavior, CF models users' taste profiles and predicts a user's preference [6]. The approach is fully exploited by the popular music website Last.fm.

Although popular, CF suffers from several drawbacks. First, it is not as scalable as content-based filtering. Thus, it may become inefficient when the total amount of user data is large. However, if the amount of user data of a system is smaller than that of music contents, CF may severely suffer from data sparsity [7]. Second, the cold start problem, aka early rater problem, is common to CF. This problem occurs to a new user because CF highly depends on users' past rating record. Third, CF has the new item problem. Newly-released albums or music have little chance to be recommended. Finally, CF makes poor predictions for users with unusual tastes since such users share little similarity with others. This is the so-called gray sheep problem [8].

3. SYSTEM AND ALGORITHM OVERVIEW

The proposed recommendation system is designed to meet two requirements. First, the user acceptance rate should be reasonably high for music recommendations based on a small amount of ratings given by the user. Second, the recommended songs should be new to the user. Recommending music unheard by a user helps the user to discover new artists and promotes hidden talents in music community. Our purpose is to build a system which satisfies the above two requirements, especially the latter one. Our system overcomes the cold start problem mentioned in section 2. The system flow chart is depicted in Figure 1. Our system uses the APIs of Last.fm to create the similar artist lists. Each component of the system is discussed in detail in this section.

3.1. Algorithm

Unlike most other systems, our system does not require a new user to provide personal information including social website account information. But our system requires the preferred artists of a user as input:

$$A = [a_1, a_2, \ldots\ldots, a_n] \tag{1}$$

where A is the preferred-artist-list of a user. For every artist a_n in A, we search for top 100 similar artists $S(a_n)$ according to the similarity score $Sim(a_n)$ retrieving from Last.fm API [9]:

$$S(a_n) = [b_{n,1}, b_{n,2}, \ldots\ldots, b_{n,100}] \tag{2}$$

$$Sim(a_n) = [sim(a_n, b_{n,1}), \ldots, sim(a_n, b_{n,100})], \tag{3}$$

$$0 \leq sim(a_n, b_{n,i}) \leq 1 \tag{4}$$

where $b_{n,m}$ is the mth similar artist. One of our purposes is to identify artists new to the user. Since artists with lower popularity are more likely to be unfamiliar to the user, they are good candidates to recommend to users. The popularity $P(x)$ of an artist x is estimated by the total result count of Google search using singer x as the key word. We can then generate a popularity list of artists similar to a_n:

$$P(a_n) = [P(b_{n,1}), P(b_{n,2}), \ldots, P(b_{n,100})]. \tag{5}$$

The estimated popularity is usually on the order of 10^3 to 10^{10}. Since a user may know artists who are more popular than those in the favorite list, we normalize the estimated popularity of an artist by the estimated popularity of the corresponding favorite artist,

$$P_{norm}(b_{n,i}) = \frac{logP(b_{n,i})}{logP(a_n)} \tag{6}$$

3.2. Key idea behind our proposed approach

People tend to like songs performed by similar artists. Thus, we set the probability of a candidate artist proportional to the similarity between the candidate artist and the corresponding favored artist. It is reasonable to assume that the novelty of a candidate artist decreases with popularity. is Given a similar artist S, we denote the likelihood of this artist to be new to the user and preferred by the user by $L(N|S)$, where N denotes new artist. The likelihood is computed by

$$L(N|S) = sim(a_n, b_{n,i}) \times (1 - P_{norm}(b_{n,i})). \tag{7}$$

Each similar artist list derived from a corresponding input favorite artist is sorted in decreasing order according to the likelihood value computed by (7). To generate a playlist (which normally is one hour long), we select songs of the top 15 similar artists from Youtube [10], one for each artist. Note that for each song, we examine whether the amount of *likes* (which is available on Youtube) is larger than *dislikes*. If not, the song is regarded as an outlier and replaced by the next popular song.

Algorithm 1 is the procedure of our system.

Algorithm 1: Automatic playlist generator

Input: A finite preferred-artists set
$A = [a_1, a_2, \ldots, a_n]$
Output: A 15-songs playlist $PL = [pl_1, pl_2, \ldots, pl_{15}]$
for $i \leftarrow 1$ ***to*** n **do**
  [S, Sim] = SearchLastfm($A(i)$)
  P_a = SearchGoogle($A(i)$)
  P = SearchGoogle(S)
  P_{norm} = $logP/logP_a$
  $L(i,:)$ = Likelihood($A(i), Sim, P_{norm}$)
end
$List$ = $argartist$SelectTop15Score(L)
PL = GetMostPopularSong($List$)
return PL
def Likelihood($A(i), Sim, P_{norm}$)
for $j \leftarrow 1$ ***to*** $|Sim|$ **do**
  LL = $Sim(1 - P_{norm})$
  Sort(LL,descend)
  return LL
end
def GetMostPopularSong($List$)
for *a in List* **do**
  s = Get most popular song on Youtube
  PL = [PL, s]
end
return PL

4. EVALUATION

To evaluate the proposed recommendation system, we recruit more than 106 subjects from campus. The subjects are asked to compare our system with Spotify Radio, which is a popular music recommendation system commercially available.

4.1. Dataset and Evaluation Methodology

Initially, each user is asked to offer 7 to 10 favorite artists as input. Each system then generates a playlist of 15 songs. We analyze the performance and robustness of the two systems through the use of a questionnaire to rate the song recommendations in terms of recommendation preference and novelty. Every subject is required to rate each song in the playlist.

Recommendation preference score is defined as the following:

- 5: The song is awesome. I will listen to it again.
- 4: Nice. I may listen to it again.
- 3: The song is okay, but I may not listen to it again.
- 2: Nothing special. No comment.
- 1: The song is terrible.

In our paper, songs that are scored above 4 are viewed as liked by users, whereas those below 2 are dislikes. Here, we exclude the songs that received a score of 3, which is neutral and does not suggest preference.

Similarly, recommendation novelty score is defined between 1 to 5:

- 5: I've neither heard of the artist nor the song.
- 4: I know the song, but I haven't heard of the singer.
- 3: I know the singer, but I haven't heard of this song.
- 2: I know both the singer and the song.
- 1: I am quite familiar with the singer and this song.

4.2. Experimental Result and Comparison

4.2.1. Recommendation Accuracy

The preference score of our system is close to the score of Spotify Radio. Figures 2 and 3 show the preference score of the two systems. In our system, 63.47% of songs receive score for 4 or 5, while 12.14% of songs received score of 1 or 2. For Spotify Radio, 59.11% of songs are liked by users, whereas 11.93% of songs are disliked. The comparison of user preference between two system is shown in Figure 4.

Score	User-preference	Novelty
5	28.89324192%	72.84313725%
4	34.57394711%	10.58823529%
3	24.38785504%	8.235294118%
2	8.814887365%	4.31372549%
1	3.33006856%	4.019607843%

Fig. 2. Experiment result of our system

Score	User-preference	Novelty
5	23.7527115%	53.2537961%
4	35.35791757%	8.459869848%
3	28.95878525%	19.19739696%
2	9.761388286%	8.242950108%
1	2.169197397%	10.84598698%

Fig. 3. Experiment result of Spotify

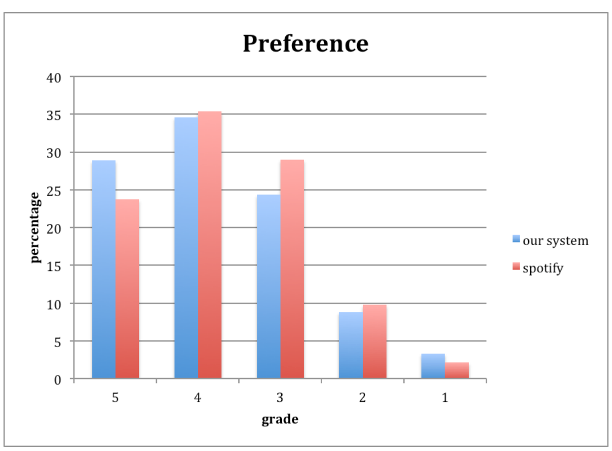

Fig. 4. Comparison of user preference between our system and Spotify

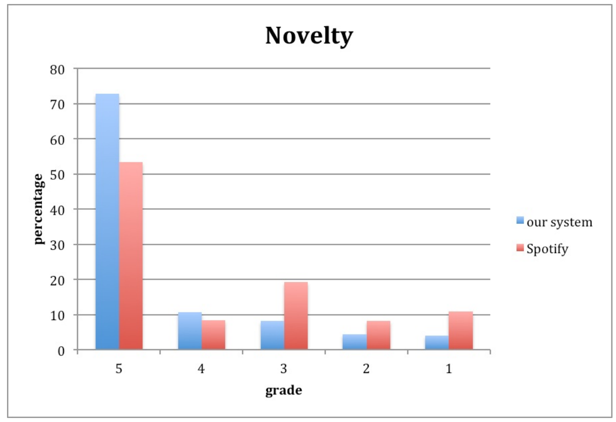

Fig. 5. Comparison of recommendation novelty between our system and Spotify

4.2.2. *Recommendation Novelty*

Figure 5 shows the novelty of the two systems. We can clearly see that our system generates songs with much higher novelty than Spotify Radio. The result indicates that 72.84% of the songs recommended by our system are totally unheard of by the users, while Spotify Radio achieves only 53.25%. The result also suggests that 19.01% of the songs provided by Spotify are songs that the users are familiar with, while it is only 8.33% for our system.

4.3. Discussion

The results show that we attain a fairly high novelty score compared to Spotify Radio and a similar preference accuracy. However, the percentage of song recommendations that received the lowest score in our system is 3.33%, which is 1.17% higher than Spotify. We believe this is the price of high recommendation novelty. But it is worth the risk because novelty is the major requirement of our system.

Besides, the overall performance of our system is still greater than that of Spotify. For the recommendation of new and favorable songs, our system achieves 55.93% whereas Spotify achieves 43.49% of all recommendation music, as shown in Figure 6.

5. CONCLUSION

In this paper, we have presented a novel music recommendation system that recommends artists who are not yet famous to users. The performance of the system is evaluated by more than 100 subjects. It achieves a high novelty score, and its recommendation accuracy is as high as the popular recommender Spotify Radio. The low complexity of the system makes it ideal for real-world applications or integration with existing music services. As future work, we plan to implement the collaborative filtering method and

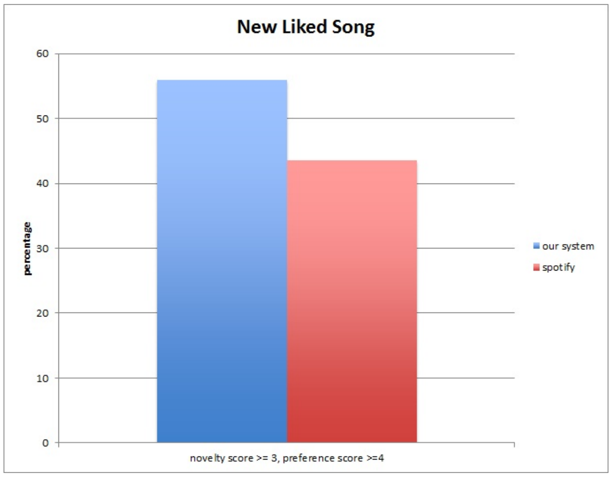

Fig. 6. Percentage of new-liked songs between our system and Spotify

integrate it with our system to address the over-specialization issue of content-based filtering.

6. REFERENCES

[1] "Spotify ltd.," `http://www.spotify.com`, [Online; accessed 2013-12-12].

[2] Apple, "Apple inc.," `http://www.apple.com/itunes/features`, [Online; accessed 2013-12-12].

[3] Michael J Pazzani and Daniel Billsus, "Content-based recommendation systems," in *The adaptive web*, pp. 325–341. Springer, 2007.

[4] Marko Balabanović and Yoav Shoham, "Fab: content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, 1997.

[5] John S Breese, David Heckerman, and Carl Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

[6] Michael A Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[7] Heung-Nam Kim, Ae-Ttie Ji, Inay Ha, and Geun-Sik Jo, "Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation," *Electronic Commerce Research and Applications*, vol. 9, no. 1, pp. 73–83, 2010.

[8] William W Cohen and Wei Fan, "Web-collaborative filtering: Recommending music by crawling the web," *Computer Networks*, vol. 33, no. 1, pp. 685–698, 2000.

[9] Last.fm, "Last.fm," `http://www.last.fm/api/`, [Online; accessed 2013-12-12].

[10] Youtube, "Youtube," `http://www.youtube.com`, [Online; accessed 2013-12-12].